Increasing Distances in Graphs

Stefan Krause

Von der Carl-Friedrich-Gauß-Fakultät für Mathematik und
Informatik der Technischen Universität Braunschweig
genehmigte Dissertation
zur Erlangung des Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.)

24. Februar 2006

Ich danke allen Angehörigen des Instituts für Mathematische Optimierung für die ausgesprochen angenehme Arbeitsumgebung, die für die Entstehung dieser Dissertation sehr wichtig war. Mein Dank gilt auch denjenigen, die durch Korrekturlesen, fachliche Anregungen und Diskussionen zum Entstehen dieser Arbeit beigetragen haben, insbesondere Jens-P. Bode, Ronny Hansmann, Professor Heiko Harborth, Sebastian Krause, Alexander Kröller, Professor Götz Uebe und Professor Uwe Zimmermann. Weiterhin möchte ich mich bei den Gutachtern bedanken, neben Professor Uwe Zimmermann sind das Professor Sándor Fekete und Professor Martin Skutella. Für die moralische Unterstützung danke ich schließlich meiner Familie und meinen Freunden.

Referent: Prof. Dr. Uwe T. Zimmermann
Korreferenten: Prof. Dr. Sándor P. Fekete, Prof. Dr. Martin Skutella
Eingereicht am 13. Oktober 2005
Prüfung am 24. Februar 2006

Contents

List of Figures

CHAPTER 1

Introduction

SHORTEST path problems arise in many areas of graph theory and combinatorial optimization. They can directly be used to model practical problems and they may occur when solving combinatorial problems like for example MAXIMUM FLOW and MAXIMUM BIPARTITE MATCHING. The effort to find a shortest path can vary in a wide range: For instances where all edges are known to have positive lengths there are fast algorithms finding a shortest path between given vertices. If all edges have the same positive length the running time can be reduced even more. However, if edges possibly have *negative lengths* and we seek *paths* rather than *walks*, that is, no vertex is visited twice, then this problem is intractable for large graphs since it includes the HAMILTONIAN PATH PROBLEM, which is *NP*-hard.

The shortest path problem studied in this work is motivated by problems in communication networks, traffic networks, and supply networks. In all of these cases one has several nodes and links connecting some of them. Usually, there will be pairs of nodes having a relatively high traffic demand. Between the nodes of such a pair one would prefer to have paths with a small number of links in order to increase the speed, decrease the response time, or to reduce the total load of the network, depending on the application. Therefore, when a network is planned one would try to find a set of links with minimum cost such that the shortest paths between the considered pairs are not longer than some bound. These problems are known as NETWORK DESIGN PROBLEMS. Many aspects of these problems have been discussed (see for example [2, 36, 41, 48]) and there is work in progress on these topics. For instance the following characteristic which is a measure for the quality of a network was considered: If links fail independently with an

individual probability what is the probability that all paths between two given nodes are disconnected? This value is called *reliability* and its determination is a hard computational problem, see [14, 46]. In this work we examine a variation where not all but only *shortest paths*, that is, paths having a minimum number of links are considered. More precisely, we ask for a minimum set of links whose failure or removal blocks all shortest paths between the given pairs of nodes. In graph theoretical terms the problem is as follows.

> **BLOCKING SHORTEST PATHS (BSP).**
> Given an undirected graph $G = (V, E)$ with edge costs c_e for all $e \in E$ and vertex pairs (s_i, t_i), $i = 1, \ldots, k$, find a minimum cost set of edges whose deletion from G blocks all shortest s_i-t_i-paths for all $i = 1, \ldots, k$.

This problem was already proposed by Bienstock and Diaz in [8], but has not been discussed in the literature until now. To avoid trivialities we can assume that there are s_i-t_i-paths for all i, otherwise some pairs s_i, t_i do not constrain the set of feasible solutions. If $\{s_i, t_i\} \in E$ then this edge forms the only shortest s_i-t_i-path and trivially has to be deleted. So we can assume that for all $i = 1, \ldots, k$ the vertices s_i and t_i are not adjacent. Furthermore, since additional edges do not affect feasibility of a solution and, if their costs are non-positive, do not increase the total cost, we can assume $c_e > 0$ for all $e \in E$.

Apparently, this problem is closely related to MINIMUM CUT and MINIMUM MULTI CUT for $k = 1$ and $k \geq 2$, respectively. Any feasible solution of these problems is also feasible for BSP, thus giving an upper bound. This bound is not tight as the instance in Figure 1.1 shows. Note that all edges are part of some shortest path. Deleting the thin edges is the unique optimal solution of BSP but any feasible solution of the corresponding MULTI CUT instance contains at least one thick edge since there is a thick s_2-t_2-path. Thus, solutions of MULTI CUT instances give inadequate upper bounds for the corresponding BSP instances. Furthermore, MINIMUM MULTI CUT is *NP*-complete for $k \geq 3$ pairs though it can be solved for $k = 2$ pairs in polynomial time, see [55]. In Chapter 4 we will explore the complexity of BSP.

There is a special case of MINIMUM MULTI CUT called MINIMUM MULTIWAY CUT where the pairs are all pairs of some set $S \subseteq V$. For $|S| = 2$ this is the simple MINIMUM CUT problem, for $|S| = 3$ this problem is *NP*-hard, see [16]. We will give some thought about the corresponding BSP instances in Section 4.3.

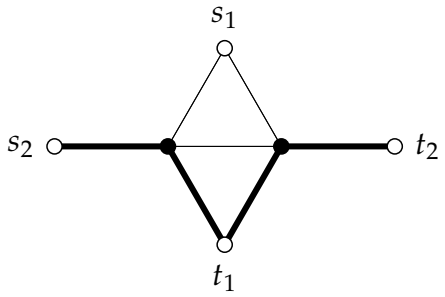

Figure 1.1. An instance with optimal objective values of 3 and $M + 2$ for BSP and MULTI CUT, respectively. Thin edges have costs 1, thick edges have costs M.

A common technique to get a lower bound for a combinatorial minimization problem is to solve the continuous relaxation of a mixed integer program (MIP) of the original problem. For BSP there is a straight forward MIP: One variable $x_e \in \{0, 1\}$ for each edge e and one inequality for each considered shortest path forcing the sum of its edge-variables to be at least 1. Then the objective function is the sum of all edge-variables x_e multiplied by their costs c_e. For the set $\mathcal{P}$ of considered shortest paths the MIP is

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e \\ \text{s. t.} \quad & \sum_{e \in P} x_e \geq 1 \quad \text{for all } P \in \mathcal{P} \\ & x_e \in \{0, 1\} \quad \text{for all } e \in E. \end{aligned}$$

Relaxing the binary constraint $x_e \in \{0, 1\}$ to $x_e \in [0, 1]$ gives a linear program (LP) whose optimal objective value is not larger then the one of the BSP instance. Although its number of inequalities may be exponential in the number of vertices it can be solved in polynomial time using ellipsoid method and polynomial time separation. Nevertheless, in Chapter 6 we will discuss another MIP for BSP whose size is bounded polynomially in the number of vertices.

For some combinatorial problems the dual linear program of the relaxation also has a combinatorial interpretation. If these two combinatorial problems have the same optimal objective value (e.g., MAXIMUM BIPARTITE MATCHING and MINIMUM BIPARTITE VERTEX COVER or MAXIMUM FLOW and MINIMUM CUT) then at least the optimal objective value can be determined efficiently by solving the LP-relaxation. For BSP with unit costs the combinatorial interpretation of the dual

LP is packing shortest paths, that is, finding a maximum set of edge-disjoint shortest paths between the considered vertex pairs. For the example in Figure 1.2 with $k = 2$ pairs the two combinatorial problems have different optimal objective values. In Chapter 7 we will use the

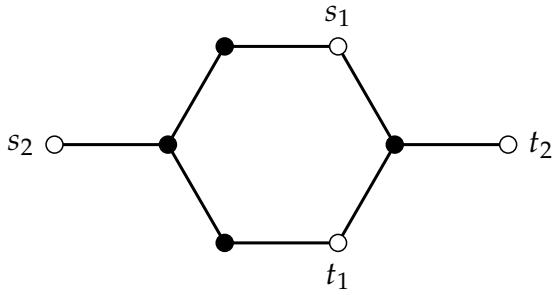

Figure 1.2. An instance of BSP with unit cost edges. The optimal objective alue is 2 but there are no two edge-disjoint shortest paths.

LP-relaxation to get an approximation algorithm for BSP, the duality gap is analyzed in Section 5.2.

Another well-known combinatorial problem related to BSP is MINIMUM SET COVER. In fact, BSP is a set cover problem with shortest paths as elements to be covered and with edges as subsets to be selected. This view enables us to apply several approximation algorithms for SET COVER to BSP, see Chapter 7.

Chapter 2 reviews some basic definitions and results in the topics graph theory, combinatorial optimization, and complexity theory used throughout this work. In Chapter 3 other combinatorial problems related to BSP are presented. Chapter 5 discusses BSP in view of extremal combinatorics. In Chapter 8 we list computational results achieved using the algorithms described in this work.

CHAPTER 2

Preliminaries

In this chapter, we list basics of the topics graph theory, complexity theory, and discrete optimization used throughout this work. We give just those definitions, notations, and results that we need. For a more detailed discussion on these subjects the reader is referred to [13, 30, 52], [3, 24, 49], and [35, 40, 44], respectively.

2.1 Basic definitions in graph theory

Here we consider *simple and finite graphs* only and therefore call them just *graphs*. They are defined as follows. A *graph* is a pair $G = (V, E)$, where $V = V(G)$ is a finite non-empty set and $E = E(G)$ consists of pairs of distinct elements of V. The elements of V are called *vertices*, the elements of E are called *edges*. If the edges are ordered pairs then G is a *directed graph* or *digraph* for short. In this case edges are written in the form (u, v) with $u, v \in V$. Otherwise G is an *undirected graph* and edges are written in the form $\{u, v\}$ with $u, v \in V$.

Graphs are depicted with dots for the vertices. For edges a continuous line between the dots corresponding to their respective vertices is used. For digraphs these lines get arrowheads at the dots of the respective second vertex of the edges, see Figure 2.1.

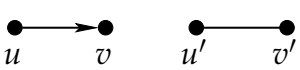

Figure 2.1. A directed edge (u, v) and an undirected edge $\{u', v'\}$.

In this work the number of vertices is denoted by n, the number of

edges is denoted by m. For the relationship between vertices and edges the following terms are common. The two vertices of an edge as well as two edges sharing one vertex are called *adjacent*, an edge and its vertices are *incident*.

Two undirected graphs $G = (V, E)$ and $G' = (V', E')$ are *isomorphic* denoted by $G \cong G'$ if there is a bijection $f : V \to V'$ such that $\{u, v\} \in E$ if and only if $\{f(u), f(v)\} \in E'$. For digraphs the definition is analogous.

Substructures of graphs are straightforward to define. For a given graph $G = (V, E)$ a graph $\widehat{G} = (\widehat{V}, \widehat{E})$ with $\widehat{V} \subseteq V$ and $\widehat{E} \subseteq E$ is a *subgraph* of G, written as $\widehat{G} \subseteq G$. If $\widehat{G}$ contains all edges of G connecting vertices of $\widehat{V}$ then $\widehat{G}$ is an *induced subgraph*.

The *complement* of a graph $G = (V, E)$ is a graph with the vertex set V and all edges that are not in E, see Figure 2.2. It is denoted by $\overline{G}$. Clearly, it holds $\overline{\overline{G}} = G$.

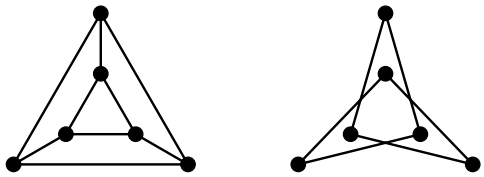

Figure 2.2. A graph G and its complement $\overline{G}$.

The *line graph* of an undirected graph $G = (V, E)$ is a graph with vertex set E and edges $\{e_1, e_2\}$ for all $e_1, e_2 \in E$ sharing a vertex in G, see Figure 2.3. There are characterizations of graphs being line graphs of

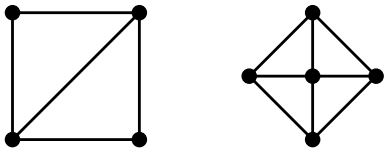

Figure 2.3. A graph and its line graph.

other graphs by means of forbidden induced subgraphs, see [42, 50, 53]
authLin
authWang.

The following local and global characteristics are frequently used in graph theory. In an undirected graph G the number of edges being incident to a vertex v is called the *degree* of v, denoted by $\deg_G(v)$ where

the subscript G may be omitted if non-ambiguous. If $\deg_G(v) = 0$ then v is called *isolated*. The smallest degree occurring in G is the *minimum degree* of G, denoted by $\delta(G)$. Analogously, the largest degree is called the *maximum degree* of G, denoted by $\Delta(G)$.

The following objects and characteristics are the ones this work deals with. For two vertices $v_1, v_2 \in V(G)$ a sequence

$$W = (v_1, u_1, u_2, \ldots, u_d, v_2)$$

of vertices of G such that any two consecutive vertices are adjacent is called a v_1-v_2-*walk*. If the vertices of a v_1-v_2-walk are distinct then it is called a v_1-v_2-*path*. A closed walk (i.e., $v_1 = v_2$) whose interior vertices all are distinct is a *cycle*. If $e \in E$ connects two consecutive vertices of W then we call e an edge of the walk W and write $e \in W$. The number of edges of a v_1-v_2-walk W is called the *length* of W. The minimum length of a v_1-v_2-walk is called the *distance* of v_1 and v_2, denoted by $\operatorname{dist}_G(v_1, v_2)$ where the subscript G may be omitted. If there is no such walk we define $\operatorname{dist}(v_1, v_2) = \infty$. It is easy to see that v_1-v_2-walks contain v_1-v_2-paths, and that therefore always a walk with minimum length being a path exists. Such a v_1-v_2-walk is called a *shortest* v_1-v_2-*path*.

The maximum distance between vertices of a graph G is its *diameter* denoted by $\operatorname{diam}(G)$. If G is not connected then $\operatorname{diam}(G) = \infty$. The *radius* of a graph $G = (V, E)$ is the number

$$\operatorname{rad}(G) = \min_{v \in V} \max_{u \in V} \operatorname{dist}_G(u, v).$$

A vertex v for which the minimum is attained is called *central*.

If for any two vertices $v_1, v_2 \in V(G)$ there is a v_1-v_2-path then G is *connected*. Maximal connected subgraphs of a graph are called *components*. For a non-trivial subset S of $V(G)$ the set of edges that are incident to exactly one vertex of S is called the *cut set* induced by S. It is denoted by $\delta(S)$. Removing all edges of a non-empty cut set from G increases the number of components. In a directed graph $\delta(S)$ contains exactly those edges (u, v) with $u \in S$ and $v \notin S$.

There are many classes of graphs with various properties. One example are graphs G whose maximum degree $\Delta(G)$ as defined before is bounded. Other common classes are given below.

- *Complete graphs* are graphs with any two vertices being adjacent, see Figure 2.4. Complete graphs with n vertices are denoted by K_n. The counterpart of complete graphs are graphs without edges, they are called *empty graphs*.

- *Cycles* are connected graphs whose vertices all have a degree of 2, see Figure 2.5. They are denoted by C_n where n is the number of vertices.

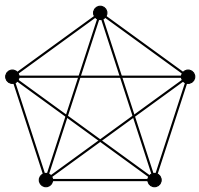

Figure 2.4. A complete graph K_5.

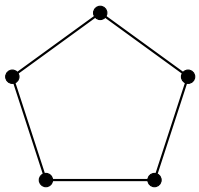

Figure 2.5. A cycle C_5.

- *Paths* are connected graphs consisting of two vertices with degree 1 and $n-2$ vertices with degree 2 where n is the number of vertices, see Figure 2.6. They are denoted by P_n.
- A *bipartite graph* is a graph whose vertices can be partitioned into two sets S and T such that all edges have one vertex in S and one vertex in T. An equivalent characterization is the following: A graph is bipartite if and only if it does not contain a cycle of odd length.
- A *complete bipartite graph* is a bipartite graph with all possible edges between S and T. It is denoted by $K_{s,t}$ where $s = |S|$ and $t = |T|$, see Figure 2.7. Complete bipartite graphs $K_{1,t}$ are called *stars*.

Figure 2.6. A path P_4.

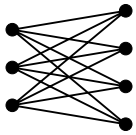

Figure 2.7. A complete bipartite graph $K_{3,4}$.

- *Planar graphs* are graphs that can be drawn in the plane such that no two lines representing edges intersect, see Figure 2.8. Such a drawing is called *planar drawing*.

Figure 2.8. The planar graph K_4 drawn with intersections (left) and without intersections (right).

There is a well-known result by Kuratowski stating that a graph is not planar if and only if it can be reduced to a K_5 or a $K_{3,3}$ by deleting edges and vertices and by replacing vertices of degree 2 together with their incident edges by single edges, see [37]. Wagner and Fáry showed independently that a planar graph can even be drawn using non-intersecting straight lines for the edges, see [19, 51].

- *Outerplanar graphs* are graphs for which a planar drawing exists such that all vertices are located on the boundary of the outer face.

  Another characterization is that a graph is outerplanar if and only if an augmentation with an additional vertex being adjacent to all other vertices leaves it planar. Starting from this condition one can adopt the characterization for planar graphs given before by replacing K_5 and $K_{3,3}$ by K_4 and $K_{2,3}$. This result is due to Chartrand and Harary, see [11].

- *Trees* are connected graphs without cycles, see Figure 2.9. It is

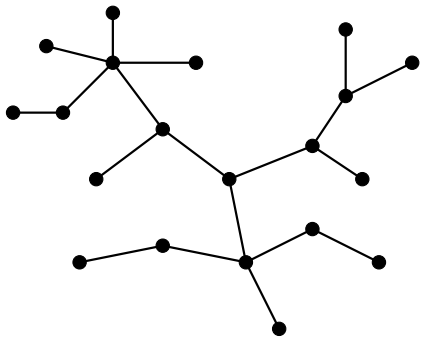

Figure 2.9. A tree.

easy to see that trees are outerplanar and bipartite. There are

many other characterizations of trees, for example:

- A graph is a tree if and only if it is connected and has n vertices and $m = n - 1$ edges.
- A graph is a tree if and only if it contains no cycle and has n vertices and $m = n - 1$ edges.
- A graph is a tree if and only if for any two vertices u and v there is exactly one u-v-path.

A vertex v of a tree with $\deg(v) = 1$ is called *leaf*. A tree that reduces to a path when all its leafs are removed is called *caterpillar*.

- *Grid graphs* are graphs that can be drawn with vertices as points with integral coordinates in the plane and edges as straight lines of unit length, see Figure 2.10. For *complete* $s \times t$ *grid graphs* there is a drawing with all points of an equidistant $s \times t$ raster and any two consecutive vertices in rows and columns being adjacent, see Figure 2.11.

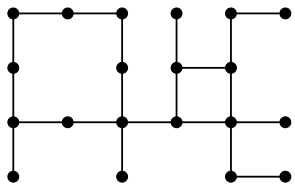

Figure 2.10. A grid graph.

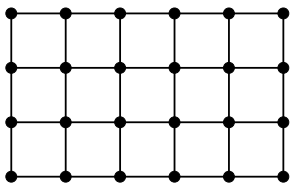

Figure 2.11. A complete 4×6 grid graph.

- *Hypercubes* Q_n are defined recursively. The graph Q_0 is one vertex and for $n \geq 1$ the graph Q_n consists of two copies of Q_{n-1} and edges between corresponding vertices, see Figure 2.12.

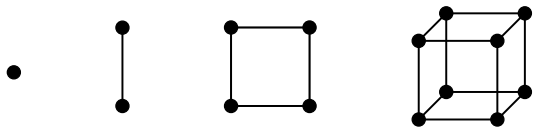

Figure 2.12. Cube graphs Q_0 through Q_3.

- *Interval graphs* are graphs for which an assignment of intervals to the vertices exists such that vertices are adjacent if and only if the corresponding intervals intersect, see Figure 2.13.
- *Circular arc graphs* are defined similar with segments of the unit circle instead of intervals, see Figure 2.14. Therefore interval graphs are also circular arc graphs.

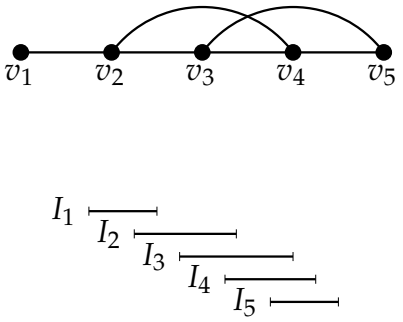

Figure 2.13. An interval graph and the intervals I_j corresponding to the vertices v_j.

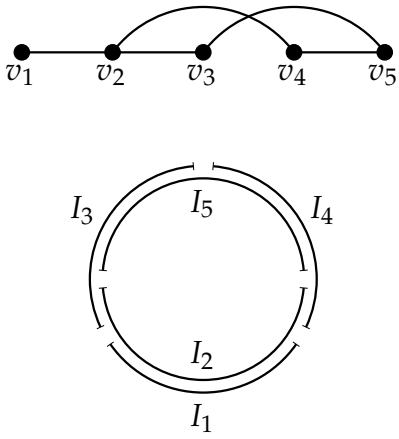

Figure 2.14. A circular arc graph and the segments I_j corresponding to the vertices v_j.

Besides these examples which will appear in the further chapters there are many other graph classes, see [10].

2.2 Short introduction to complexity theory

In complexity theory one asks for the *effort* to solve problems of a specific type dependent on their *size*. The actual running time of an algorithm is up to the hardware, and even the required memory in bytes may vary on different computer architectures. Therefore we only ask for the growth, and state that the time or space used for computation is $\mathcal{O}(f(n))$ if it is bounded from above by $cf(n)$ for large n, where $f(n)$ is some explicitly given expression dependent on the size n of the input data and c is some constant. If an algorithm accesses every byte

of the necessary memory at least once then the computation time will dominate the memory requirements, hence we will focus on the time expense only.

We start with the class of decision problems which are pleasant in terms of required computation time.

Definition 2.1
The class P consists of those decision problems that can be solved in polynomial time.

To prove that a problem is in P one has to give a polynomial time algorithm. If for some problem no such algorithm is known although much research effort has been expended, one might tempt to conjecture that this problem is not in P. Admittedly, the non-existence of an algorithm is hard to prove.

The second class of problems we define can be solved in polynomial time by a *non-deterministic algorithm*, that is, an algorithm which guesses (part of) the solution and checks its feasibility until success. Clearly, the running time of such an algorithm is not deterministic and therefore we may ask if there is a positive probability that it solves the decision problem in polynomial time.

Definition 2.2
The class NP consists of those decision problems that can be solved by a non-deterministic algorithm in polynomial time.

So problems in *NP* have the property that a "yes"-answer can be verified in polynomial time. A well-known problem in *NP* which also plays a central role in the theory of P and *NP* is the following.

Definition 2.3
Let be given a set $X = \{x_1, \ldots, x_n\}$ of boolean variables and a set of clauses each of them being the disjunction of some elements of X and negated elements of X. The problem of deciding whether there is an assignment to the variables such that the conjunction of these clauses is true is called SATISFIABILITY PROBLEM *or SAT for short.*

An example of an instance of SAT is

$$(x_1 \vee \neg x_2 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_3 \vee \neg x_4).$$

A non-deterministic algorithm for SAT is for example the following: select the values for the variables randomly and check whether the overall expression is true. Both parts can be carried out in polynomial time

and since only a finite number of assignments for the variables exist, there is a positive probability to guess a valid solution.

If a decision problem can be solved by a deterministic polynomial algorithm then a guessed solution can be checked in polynomial time as well, therefore $P \subseteq NP$. Whether or not $P \neq NP$ holds is probably the most popular open problem in optimization and computer science. To prove inequality one would consider problems that are at least as hard to solve as all other problems in NP. To prove this property we transform instances of a problem Π_1 to instances of a problem Π_2 in polynomial time such that corresponding instances are equivalent. If this is possible then we say that Π_1 can be *reduced* polynomially to Π_2. Candidates to be in $NP \setminus P$ the following problems.

Definition 2.4
*A problem $\Pi \in NP$ is called NP-*complete *if every problem in NP can be* reduced *to Π in polynomial time, that is, a polynomial algorithm for Π would lead to polynomial algorithms for all problems in NP.*

So NP-complete problems are hardest to solve within the class NP. In 1971 Cook proved in [15] that such a problem exists.

Theorem 2.1 (Cook)
SAT is NP-complete.

Using this result one can prove that a problem $\Pi \in NP$ is NP-complete by showing that SAT (or any other NP-complete problem) can be reduced to Π in polynomial time. If SAT can be reduced polynomially to Π, but it is not clear whether or not $\Pi \in NP$ holds, then Π is called *NP-hard*.

There is a strong link between decision problems and optimization problems of the general form $\min\{f(x) \mid x \in S\}$. If we have a lower bound l and an upper bound u on the optimal objective value then we can bisect this range by solving a decision problem of the form "does $\min\{f(x) \mid x \in S\} < \frac{1}{2}(l+u)$ hold?". Thus, if the decision problem can be solved in polynomial time and the bounds are at most exponential in the size of the input data, then we can solve the optimization problem up to some accuracy in polynomial time by iterative bisection. Vice versa, if the optimization problem can be solved polynomially then this is true for the decision problem as well. So if the decision problem is NP-complete then it is likely that the optimization problem is hard to solve. In this case one may ask for algorithms that determine non-optimal but *good* solutions in polynomial time.

Definition 2.5
Let a minimization problem Π and a polynomial time algorithm A be given. Denote the objective values of the solution of an instance $I \in \Pi$ determined by A and of the optimal solution by $z_A(I)$ and $z_{\text{opt}}(I)$, respectively. If for any $I \in \Pi$ it holds

$$z_A(I) \leq r(|I|) z_{\text{opt}}(I)$$

then A is called $r(|I|)$-approximation algorithm for Π, *where $r : \mathbb{N} \to \mathbb{R}$ is the* performance guarantee *and $|I|$ denotes the* size *of the instance (e.g., the amount of memory required to store I). If r is a constant function then A is called a* constant factor approximation. *The class of problems allowing a constant factor approximation is called APX which is an abbreviation for approximable.*

Approximation algorithms and performance guarantees for maximization problems can be defined similar. If the considered problem is hard to solve then the best thing one can hope for is a sequence of approximation algorithms such that the sequence of their performance guarantees converges to 1. Such sequences and the class of problems for which such sequences exist are defined as follows.

Definition 2.6
A set of algorithms containing an r-approximation for each fixed r is called a polynomial time approximation scheme. *The class PTAS consists of the problems for which a polynomial time approximation scheme exists.*

Clearly it holds $PTAS \subseteq APX$ but there are problems in APX which are not in $PTAS$ unless $P = NP$. One well-known example is VERTEX COVER where we seek a minimum set S of vertices such that each edge is adjacent to at least one vertex of S. There is a simple 2-approximation for this problem by iteratively selecting vertex disjoint edges as long as possible and then returning both vertices of all these edges. However, for $r < 7/6$ there is no r-approximation for this problem unless $P = NP$ as Håstad showed in [32]. Problems like MINIMUM VERTEX COVER which are in APX but which are not in $PTAS$ unless $P = NP$ are called *APX-hard*.

Finally, the question arises whether there are optimization problems which are not contained in APX provided that $P \neq NP$. One example is the famous TRAVELING SALESMAN PROBLEM (TSP) which asks for a cycle being minimal with respect to given edge weights and

containing each vertex of the given graph. If the triangle inequality holds for the edge weights then the algorithm by Christofides determines a solution whose objective value is at most $\frac{3}{2}$ as large as the optimal value, see [12]. Nevertheless, for the general case the existence of a constant factor approximation would imply $P = NP$.

2.3 Topics in discrete optimization

In discrete optimization one has a characterization of some finite set and asks for an element of this set being optimal with respect to some objective function. Probably the most general discrete problem is the following. For its definition we denote the set of non-negative integers by $\mathbb{Z}_+$ and the set of non-negative rational numbers by $\mathbb{R}_+$.

Definition 2.7
A problem of the form

$$\max \left\{ c^T x \;\middle|\; Ax \leq b,\ x \in \mathbb{Z}_+^r \times \mathbb{R}_+^{n-r} \right\} \tag{2.1}$$

with $c \in \mathbb{Q}^n$, $A \in \mathbb{Q}^{m \times n}$, $b \in \mathbb{Q}^m$ *and* $1 \leq r \leq n$ *is called* MIXED INTEGER PROGRAM *abbreviated as MIP.*

It is easy to see that even deciding whether there is a feasible solution for (2.1) is *NP*-hard since SAT can be formulated straightforward as a MIP. Despite its hardness MIPs sometimes are a good approach to solve combinatorial problems and in some cases they give new insights using the following objects. An important tool to solve MIPs is defined below.

Definition 2.8
For the system in (2.1) the linear program (LP)

$$\max \left\{ c^T x \;\middle|\; Ax \leq b,\ x \in \mathbb{R}_+^n \right\} \tag{2.2}$$

is called its LP-relaxation.

Since the set in (2.2) is a superset of the one in (2.1) the maximum of the former is not smaller than the maximum of the latter. Therefore the LP-relaxation, which can be solved in polynomial time using ellipsoid method or interior point method, gives an upper bound on a MIP, that is,

$$\max \left\{ c^T x \;\middle|\; Ax \leq b,\ x \in \mathbb{Z}_+^r \times \mathbb{R}_+^{n-r} \right\} \leq \max \left\{ c^T x \;\middle|\; Ax \leq b,\ x \in \mathbb{R}_+^n \right\}.$$

If LPs as in (2.2) have an optimal solution then they also have an optimal *basic solution*, that is, a solution that is not located on a line joining two other feasible points. When solving an LP one usually determines such a solution, for instance the widely used simplex algorithm examines basic solutions only.

As we will see later on, for some combinatorial problems basic solutions of the LP-relaxation of its MIP formulation always are integral. Consequently, optimal solutions of the relaxation are also optimal for the given problem. Therefore it suffices to determine optimal basic solutions of the relaxation, which can be done in polynomial time using ellipsoid method or interior point method as long as the size of the MIP is bounded polynomially by the size of the combinatorial problem. If the number of inequalities of the integral LP-relaxation grows exponentially with the size of the input data, then one needs a so-called *polynomial time separation algorithm*, that is, an algorithm that determines an inequality violated by a given point x in polynomial time. Using this separation algorithm and the ellipsoid method an LP can be solved in polynomial time despite of its exponential size.

For some problems the well-known duality in linear programming allows a combinatorial interpretation, too.

Definition 2.9
The linear program

$$\min \left\{ b^T y \;\middle|\; A^T y \geq c,\ y \in \mathbb{R}^m_+ \right\} \tag{2.3}$$

is called the dual linear program *of the system in (2.2).*

Note that the dual LP of (2.3) is again the system (2.2) justifying the term "duality". If one of the systems (2.2) or (2.3) has an optimal solution then this is also true for the other one and the optimal objective values are the same, that is,

$$\max \left\{ c^T x \mid Ax \leq b,\ x \in \mathbb{R}^n_+ \right\} = \min \left\{ b^T y \mid A^T y \geq c,\ y \in \mathbb{R}^m_+ \right\}. \tag{2.4}$$

A well-known example of this result is the well-known MAX FLOW-MIN CUT theorem. The primal problem is the following.

> **MAXIMUM FLOW.**
> Given a directed graph $G = (V, E)$, a source vertex $s \in V$, a sink vertex $t \in V$, and a non-negative integral capacity

function $c : E \to \mathbb{Z}_+$, we seek an integral flow, that is, a function $x\colon E \to \mathbb{Z}_+$ with $x(e) \leq c(e)$ for all $e \in E$ and

$$\sum_{e \in \delta(\{v\})} x(e) = \sum_{e \in \delta(\overline{\{v\}})} x(e)$$

for all $v \in V \setminus \{s,t\}$ such that $\sum_{e \in \delta(\{s\})} x(e)$ is maximal.

Formulating this problem as a MIP and considering the dual of the LP-relaxation leads to the following problem.

MINIMUM DIRECTED CUT.
Given a directed graph $G = (V,E)$, a source $s \in V$, a sink $t \in V$, and a non-negative integral cost function $c\colon E \to \mathbb{Z}_+$, we seek an edge set S with minimum total cost such that $\mathrm{dist}_{G \setminus S}(s,t) = \infty$.

For both problems there are MIP formulations whose relaxations have integral optimal solutions. Since these relaxations are mutually dual they have the same optimal objective value. This result was proved by Ford and Fulkerson (see [21]) using combinatorial arguments rather than LP duality. Consequently, combinatorial algorithms for both problems are known that are not based on LPs.

The MINIMUM DIRECTED CUT problem can also be formulated for undirected graphs and is called MINIMUM CUT. Generalizations of this problem are given below.

MINIMUM MULTI CUT.
Given an undirected graph $G = (V,E)$ with edge costs c_e for all $e \in E$ and vertex pairs (s_i,t_i), $i = 1,\ldots,k$, find a set S of edges with minimum total cost such that $\mathrm{dist}_{G \setminus S}(s_i,t_i) = \infty$ for all $i = 1,\ldots,k$.

This problem is *NP*-complete for $k \geq 3$ but polynomially solvable for $k = 2$, see [55]. The following problem is another generalization of MINIMUM CUT.

MINIMUM MULTIWAY CUT.
Given an undirected graph $G = (V,E)$ with edge costs c_e for all $e \in E$ and vertices (so-called terminals) s_i, $i = 1,\ldots,k$, find a set S of edges with minimum total cost such that $\mathrm{dist}_{G \setminus S}(s_i,s_j) = \infty$ for all i and j with $1 \leq i < j \leq k$.

For $k = 2$ this problem is identical to MINIMUM CUT and for $k \geq 3$ it is *NP*-complete, see [16].

Another well studied combinatorial problem related to this work is the problem of finding a shortest s-t-path in a graph $G = (V, E)$ with $s, t \in V$. A simple algorithm for this purpose is the following.

Algorithm 2.1 (breadth-first-search)
for $i \in V$ do
  $N(i) \leftarrow \emptyset$;
for $(u, v) \in E$ do
  $N(u) \leftarrow N(u) \cup \{v\}$;
  $N(v) \leftarrow N(v) \cup \{u\}$; (for undirected graphs only)
for $i \in V$ do
  $d(i) \leftarrow \infty$;
$d(s) \leftarrow 0$;
$p(s) \leftarrow -1$;
queue $Q \leftarrow (s)$;
while Q is not empty do
  extract and delete v from front of Q;
  for $i \in N(v)$ do
    if $d(i) = \infty$ then
      $d(i) \leftarrow d(v) + 1$;
      $p(i) \leftarrow v$;
      append i to back of Q;

Since the number of edges normally dominates the number of vertices the running time of this algorithm is $\mathcal{O}(m)$. After applying this algorithm $d(i)$ contains the number $\operatorname{dist}(s, i)$ and if $d(i) \neq \infty$ the vertices of a shortest s-i-path can be determined in reverse order by

$$i,\ p(i),\ p(p(i)),\ \ldots,\ p(p(\cdots p(i) \cdots)) = s.$$

If we do not seek s-t-paths having a minimal number of edges but want s-t-paths having a minimal sum of non-negative edge weights w_e then the following algorithm by Dijkstra ([17]) can be used.

Algorithm 2.2 (Dijkstra's algorithm)
for $i \in V$ do
  $N(i) \leftarrow \emptyset$;
for $(u, v) \in E$ do
  $N(u) \leftarrow N(u) \cup \{v\}$;
  $N(v) \leftarrow N(v) \cup \{u\}$; (for undirected graphs only)

```
for i ∈ V do
    d(i) ← ∞;
    t(i) ← 0;
d(s) ← 0;
p(s) ← s;
for ℓ = 1 to |V| do
    i ← −1;
    for j ∈ V do
        if t(j) = 0 then
            if i = −1 or d(j) < d(i) then
                i ← j;
    if d(i) = ∞ then
        stop;                                   (graph is not connected)
    for j ∈ N(i) do
        if t(j) = 0 and d(j) > d(i) + w_(i,j) then
            d(j) ← d(i) + w_(i,j);
            p(j) ← i;
    t(i) ← 1;
```

In this simple implementation the running time is $\mathcal{O}(n^2)$ but it can be improved using special data structures. Similar to Algorithm 2.1 $d(i)$ contains the minimum weight of s-i-paths with respect to the edge weights w_e. The paths can be determined using $p(i)$ as described above.

Dijkstra's algorithm does not work if some weights w_e are negative. If at least no cycle having a negative sum of weights exists, one can use the algorithm by Bellman and Ford to find minimum weight s-t-paths, see [6, 20]. If such cycles may exist then this problem is *NP*-hard.

CHAPTER 3

Related Problems

THERE are some well-known problems that are evidently related to BSP, namely MINIMUM CUT, MINIMUM MULTI CUT, and MINIMUM MULTIWAY CUT. These obvious examples were already mentioned in the introduction and will appear many times in this work. The difference between them and BSP is that solutions for the latter block shortest paths only. In this chapter we present other problems and results concerned with affecting distances and other graph properties by modifying graphs.

3.1 Results on edge deletion problems

The general edge deletion problem is an optimization problem of the following kind.

> **EDGE DELETION PROBLEM.**
> Given a graph G, find a minimum cardinality or minimum cost edge set whose deletion from G results in a subgraph H having some property.

Many problems in graph theory and combinatorial optimization can be expressed as edge deletion problems. Besides the well-known examples mentioned at the beginning of this chapter there are other problems that in fact are edge deletion problems. Determining a minimum cost spanning tree, a minimum cost matching, or a minimum cost solution of the TRAVELING SALESMAN PROBLEM can be transformed to maximization problems by simply negating the edge costs. Then one can ask for the minimum cost edge set whose deletion results in a tree, a

matching, or a cycle on n vertices, where n is the number of vertices of G. As these examples show, some edge deletion problems may be solvable in polynomial time while others seem to be intractable.

What is a motivation to consider edge deletion problems? Think of an optimization problem that is *NP*-complete in general and solvable in polynomial time on graphs of a special class $\mathcal{H}$. If we find a graph $H \in \mathcal{H}$ *similar* to the given graph G then an optimal solution of the problem on H may lead to a *good* solution of the problem on G. For instance you may want to determine the chromatic number $\chi(G)$ of a graph G, that is, the smallest number c such that a coloring of the vertices of G with c colors exists where adjacent vertices have different colors. It is well-known that this problem is *NP*-hard. However, it can be solved in polynomial time for *perfect graphs*, that is, graphs whose induced subgraphs G' all contain a complete subgraph with $\chi(G')$ vertices, see [28]. So you may seek a perfect subgraph H of G having maximum number of edges, determine a minimal coloring of H in polynomial time, and try to modify colors of those vertices being incident to the removed edges to get a coloring of G with a *small* number of colors. Unfortunately, finding a maximum perfect subgraph of G is *NP*-hard itself since this includes testing whether G is perfect.

There is a wide range of publications concerning edge deletion problems and many edge deletion problems are known to be *NP*-hard, for instance those with one of the properties

(i) without cycles of length ℓ for fixed $\ell \geq 3$,

(ii) without cycles of length at most ℓ for fixed $\ell \geq 3$,

(iii) connected and with bounded maximum degree,

(iv) outerplanar,

(v) planar,

(vi) bipartite

for H (see [18, 54]).

3.2 Vertex deletion problems

The problem of blocking shortest paths by vertex deletion (BSPv) can be defined like BSP, the only difference is that vertices are deleted instead

of edges. Needless to say, vertices of the considered pairs may not be deleted. The two problems are closely related: Any instance of BSP can be transformed to an instance of BSPv in the following way. For each pair (s_i, t_i) we insert vertices s_i', t_i' and edges $e_i^s = \{s_i, s_i'\}$ and $e_i^t = \{t_i, t_i'\}$. Now we consider the line graph and the problem of blocking all shortest paths between the vertices e_i^s and e_i^t for all $i = 1, \ldots, k$ by vertex deletion. It is easy to see that shortest s_i-t_i-paths in the original graph correspond exactly to shortest e_i^s-e_i^t-paths in the line graph. Therefore the instance of BSP and the instance of BSPv are equivalent. As we will see in Chapter 4 it is *NP*-hard to solve BSP even for unit costs and $k = 2$. Since the just described reduction is polynomial, BSPv is also *NP*-hard in line graphs with unit costs and $k = 2$. Note that the reverse reduction is not possible since for a general BSPv instance deleting a vertex from the graph corresponds to deleting the edges of a clique of its line graph instead of deleting single edges of its line graph.

BSPv is one example of the so-called *vertex deletion problems* which can be discussed in a more general way as Lewis and Yannakakis did in 1980, see [38]. More precisely, they considered the problem given below.

> **VERTEX DELETION PROBLEM.**
> Given a graph G find a set of vertices of minimum cardinality, whose deletion from G results in a subgraph satisfying some given property π.

They showed that this problem is *NP*-hard under the following assumptions, even when restricted to planar graphs.

(*i*) The property π is *hereditary on induced subgraphs*, that is, if a graph G has property π, then all vertex-induced subgraphs of G have property π as well.

(*ii*) The property π is *non-trivial*, that is, there are infinitely many graphs having property π as well as infinitely many graphs not having π.

Examples for properties π satisfying these assumptions are outer planar, bipartite, interval graph, complete, and empty. For the last one the vertex deletion problem is equivalent to the well-known MINIMUM VERTEX COVER. Over the recent years there has been progress in approximating vertex deletion problems, see [22, 23, 39].

3.3 Changing diameter or radius of graphs

Besides the general approaches to edge deletion problems described in Section 3.1 non-algorithmic results on increasing the diameter or the radius (see page 7) by edge deletion are known. For instance Schoone, Bodlaender, and van Leeuwen asked for the largest diameter of a connected graph H obtained from a graph G with diameter D by deleting k edges, see [43]. They proved that if $\operatorname{diam}(H)$ is maximum and H is connected, then

$$(k+1)D - k \leq \operatorname{diam}(H) \leq (k+1)D$$

for even D and

$$(k+1)D - 2k + 2 \leq \operatorname{diam}(H) \leq (k+1)D$$

for odd $D \geq 3$. Furthermore, for directed graphs G they proved

$$\operatorname{diam}(H) = \left\lfloor \tfrac{1}{2} + \sqrt{2k + \tfrac{1}{4}} \right\rfloor \quad \text{and} \quad \operatorname{diam}(H) = 2k + 2$$

for $D = 1$ and $D = 2$, respectively, while for $D \geq 3$ there is no bound on $\operatorname{diam}(H)$ in terms of k and D. Additionally they showed that it is *NP*-complete to decide whether k edges can be deleted from a given graph G such that the resulting graph has a specific diameter. Both problems are hard in directed as well as in undirected graphs. They also proved the similar problem to be *NP*-hard where the diameter is decreased to a given value by edge insertion.

In a subsequent work Alon, Gyárfás, and Ruszinkó considered the minimal number $f_d(G)$ of edges that have to be added to a graph G such that the resulting graph has a diameter of at most d, see [1]. They verified that $f_2(G) = n - \Delta(G) - 1$ and $f_3(G) = n - \mathcal{O}(\Delta(G)^3)$ hold if the number n of vertices is sufficiently large, where $\Delta(G)$ is the maximum degree of G, see page 7. For $d \geq 4$ they proved the tight bound $f_d(G) = n/\lfloor d/2 \rfloor - \mathcal{O}(1)$.

Similar questions may be asked for the radius of a graph. Segawa showed in [45] that if H is a connected graph obtained from a graph G by removing k edges then it holds

$$\operatorname{rad}(H) \leq (k+1)\operatorname{rad}(G) - \left\lfloor \tfrac{k}{2} \right\rfloor,$$

where $\operatorname{rad}(G)$ denotes the radius of G.

Graham and Harary considered in [26] a special case of diameter alteration by restricting to the hypercubes Q_n (see page 10). They figured out that one has to remove at least $n-1$ edges from Q_n to increase its diameter, that at least two edges must be added to Q_n to decrease its diameter, and that both bounds are tight. Then they discussed the corresponding maximum values and proved that the maximum number of edges that can be added to Q_n without decreasing its diameter is

$$\binom{2n}{n-1} + \frac{1}{2}\binom{2n}{n} - (n+1)2^{n-1}.$$

For the maximum number of edges which one can remove from Q_n without increasing its diameter they gave bounds that were improved by Bouabdallah, Delorme, and Djelloul in [9] to

$$(n-2)2^{n-1} - \binom{n}{\lfloor n/2 \rfloor} + 2$$

and

$$(n-2)2^{n-1} + 1 - \left\lceil \frac{2^n - 1}{2n-1} \right\rceil$$

for the lower and the upper bond, respectively.

3.4 Decreasing distances by edge insertion

Since inserting edges into a graph G is equivalent to deleting edges from its complement $\overline{G}$, one could think that the problem of changing distances by edge insertion is closely related to the problems mentioned before. However, either $\operatorname{dist}_G(u,v) = 1$ or $\operatorname{dist}_{\overline{G}}(u,v) = 1$ holds trivially. Also for the diameter only few combinations for $\operatorname{diam}(G)$ and $\operatorname{diam}(\overline{G})$ are possible. In [31] Harary and Robinson proved that if $\operatorname{diam}(G) \geq 3$ then $\operatorname{diam}(\overline{G}) \leq 3$. In fact, exactly

$$\begin{aligned}
&\operatorname{diam}(G) = 1, \quad \operatorname{diam}(\overline{G}) = \infty, \\
&\operatorname{diam}(G) = 2, \quad \operatorname{diam}(\overline{G}) = d \text{ for all } d \geq 2, \\
&\operatorname{diam}(G) = 3, \quad \operatorname{diam}(\overline{G}) = 3
\end{aligned}$$

are possible. Therefore it is unlikely that results or algorithms for a deletion problem are adaptable for an insertion problem or vice versa and the problem given below is independent of BSP.

DISTANCE DECREASING PROBLEM.
Given an undirected graph $G = (V, E)$, costs c_e for all edges of $\overline{G}$, and vertex pairs (s_i, t_i), $i = 1, \ldots, k$, find a set of edges of $\overline{G}$ with minimum total cost whose insertion into G decreases $\text{dist}_G(s_i, t_i)$ for all $i = 1, \ldots, k$ by at least 1.

This problem seems to be undiscussed. Nevertheless, it is easy to see that it is *NP*-hard and *APX*-hard. The proof is by a simple reduction from the MINIMUM STEINER TREE problem which is defined as follows.

MINIMUM STEINER TREE PROBLEM.
Given a complete graph $K_n = (V, E)$, positive edge weights c_e for all $e \in E$, and a subset $S \subseteq V$ of required vertices, find a minimum cost subtree of K_n containing all vertices in S.

This problem is *NP*-hard and *APX*-hard, see [7]. It can be solved or approximated using the following instance of the problem described before. The graph is the empty graph $G = \overline{K_n}$, the edge weights of $\overline{G} = K_n$ are the given weights, and the pairs are all pairs of vertices of S. It is easy to see that a subset of the edges of the original graph is a feasible solution of this DISTANCE DECREASING PROBLEM if and only if it is feasible for the MINIMUM STEINER TREE PROBLEM. Therefore optimal objective value and approximation ratio are preserved. If the graph of the DISTANCE DECREASING PROBLEM shall be connected one could join any two vertices of G by paths using $|V|$ additional vertices each. The additional edges in G get sufficiently large weights, for example the sum of the given weights.

Many results on Steiner trees are known and many variants of this question have been discussed, see for instance [34]. Probably some of the results are applicable to this edge insertion problem.

3.5 Flows with restricted path length

In his Ph. D. thesis [4] Baier discusses a problem related to BSP. As a motivating example he gives the following traffic problem: Besides the fact that all road users should be able to reach their destination, a limitation of detours is desirable. Therefore the flow corresponding to the overall traffic should consist of relatively short paths representing the individual routes of the passengers. This leads to *length-bounded*

flows, that is, flows that can be decomposed into paths of restricted length. The combinatorial dual problem of BSP was presented as packing shortest paths in the introduction and it will be discussed more detailed in Section 5.2. In fact, Baier considers a generalization of this problem where not only shortest paths, but paths up to a given length are packed. Moreover, edges do not necessarily have unit length as in this work. Baier shows that even the fractional relaxation, that is, a rational flow value is assigned to the paths, with one source-sink-pair is *NP*-hard, even in outerplanar graphs. Furthermore he discovers that for given flow values on the edges, it is *NP*-hard to find a decomposition into paths of restricted length.

Since this special flow problem is already hard to solve for $k = 1$ pair this is true for non-fixed $k \in \mathbb{N}$ as well. However, as Baier proves, the general case can be solved by determining solutions of a polynomially bounded set of instances with $k = 1$ pair only.

As far as approximation is concerned Baier gives two fully polynomial time approximation schemes. The first one approximates the length bound, that is, the optimal flow value is achieved using path-flows whose lengths are at most $(1 + \varepsilon)$ times the allowed length. The second algorithm approximates the optimal flow value by $1/(1 + \varepsilon)$ while meeting the length bound. Both approaches rely on general linear programming techniques and therefore no meaningful estimation of its running time can be given.

Furthermore, Baier discusses *length bounded s-t-cuts*, that is, edge sets containing an edge of all s-t-paths up to some given length. Clearly, this is a generalization of BSP. We give one of his main results in Chapter 9.

CHAPTER 4

Results on Complexity

In this chapter we discuss the complexity of BSP and several special cases of it. The target is to find the borderline between the *easy* and the *hard* cases and to get an idea of what it is that makes this problem hard.

4.1 Single pair and star instances

The simplest case where the shortest paths of only one pair are to be blocked can be solved in polynomial time. For this purpose we need a special digraph which is defined as follows.

Definition 4.1
Let a simple and undirected graph $G = (V, E)$ be given. Then for $s \in V$ the digraph $D_s = (V, A)$ with

$$A = \{(u, v) \mid \{u, v\} \in E \quad \textit{and} \quad \operatorname{dist}_G(s, v) = \operatorname{dist}_G(s, u) + 1\}$$

is called the shortest paths digraph *with root s. It is denoted by D_s.*

Figure 4.1 shows a generalized Petersen graph and its shortest paths digraph. As a consequence of the symmetry of this special graph, all digraphs D_s with s being a vertex of the inner cycle are isomorphic.

By definition the vertices of any directed s-t-path in D_s have strictly increasing distances from s in G. Thus, any directed s-t-path in D_s corresponds to a shortest s-t-path in G. Vice versa each edge of any shortest s-t-path in G occurs with appropriate orientation in D_s, so any shortest s-t-path in G corresponds to a directed s-t-path in D_s. Therefore, solving BSP for a graph G and one vertex pair (s, t) is equivalent

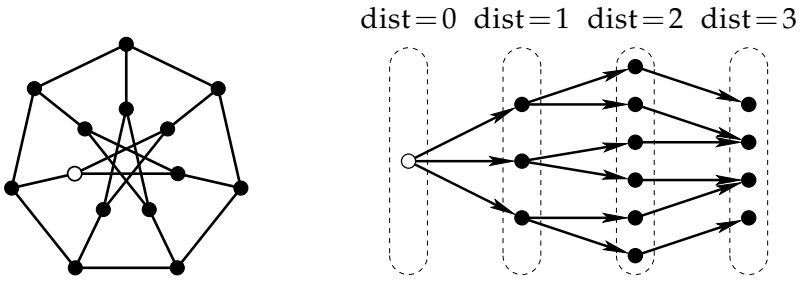

Figure 4.1. A graph and its shortest paths digraph with the white vertex as root.

to determining a minimum cost s-t-cut in the shortest paths digraph where the edge costs in the digraph are the same as in G. The next theorem states that this result can be extended to so-called *star instances*, that is, instances where the pairs have a star-like configuration.

Theorem 4.1
Instances of BSP with pairs (s, t_i), $i = 1, \ldots, k$, that is, one vertex occurs in all pairs, can be solved in polynomial time.

PROOF. Since the shortest paths digraph depends on its root only, directed s-t_i-paths in D_s correspond exactly to shortest s-t_i-paths in G ($i = 1, \ldots, k$), as described above. So an optimal solution of the BSP instance can be found by determining a minimum cost edge set in D_s separating s and the set $\{t_i \mid i = 1, \ldots, k\}$. It is well-known that this can be done by introducing a *super sink* t and directed edges (t_i, t) with sufficiently large costs (e.g., $1 + \sum c_e$) and then computing a minimum s-t-cut. ❑

4.2 Instances with two disjoint pairs

Yannakakis, Kanellakis, Cosmadakis, and Papadimitriou showed in [55] that MULTI CUT with two pairs can be solved in polynomial time. This is not true for BSP, even for grid graphs as the following theorem shows.

Theorem 4.2
BSP with $k = 2$ pairs (s_i, t_i) is NP-hard, even for grid graphs G.

PROOF. Whether a given set of edges is a feasible solution of BSP can be checked in polynomial time. Thus, this problem belongs to *NP*. We

prove the completeness by reduction from SET COVER. So let non-empty sets $A_1, \ldots, A_m \subseteq \{1, \ldots, n\}$ and $r \in \{1, \ldots, m\}$ be given. To decide whether there is an index set I with $|I| \leq r$ and $\bigcup_{i \in I} A_i = \{1, \ldots, n\}$ we construct a grid graph as an instance of BSP and ask for a solution with total cost of at most c (the value $c = c(r)$ will be specified later).

We take a tiling with m rows and n columns. In each cell we place one of the two types of tiles shown in Figure 4.2. The single thin edge

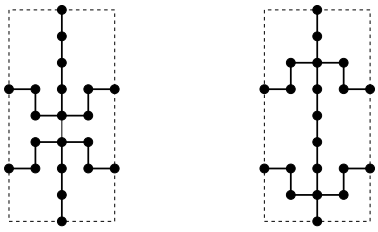

Figure 4.2. Tiles of type 1 (left) and type 2 (right).

in the type 1 tile has cost 1 whereas thick edges have costs $c + 1$, so they cannot be deleted. These tiles both have two connector vertices on the right and the left side and one connector vertex on the top and the bottom side shared with neighboring tiles. For the cell in the ith row and the jth column we choose a tile of type 1 if $j \in A_i$ and a tile of type 2 otherwise. Then in each row we add one vertex $\overline{s}_i$ to the left of the tiling and connect it to the upper connector vertex on the left side of the leftmost cell by an edge with cost $|A_i| - 1$ where i is the number of the row (see Figure 4.3). The lower connector vertices on the right

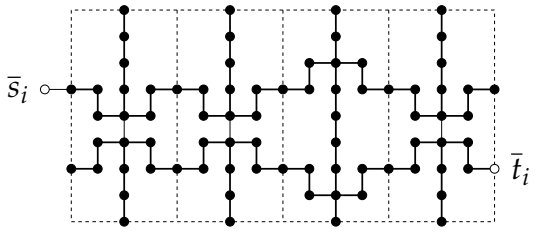

Figure 4.3. Example for the ith row in the reduction from SET COVER.

side of the rightmost cells are denoted by $\overline{t}_i$, where i is the number of the row. The vertices on the top side and on the bottom side of the tiling are denoted by $\widehat{s}_j$ and $\widehat{t}_j$, respectively, where j is the number of the column.

In this graph shortest $\bar{s}_i$-$\bar{t}_i$-paths use the edge adjacent to $\bar{s}_i$, six edges from left to right in each cell, and exactly one additional vertical edge in a type 1 cell, so their length is $6n+2$. To disconnect all shortest paths of an $\bar{s}_i$-$\bar{t}_i$-pair one can delete the edge being adjacent to $\bar{s}_i$ with cost $|A_i|-1$ or one can delete all $|A_i|$ unit cost edges in the type 1 cells in that row.

Shortest $\widehat{s}_j$-$\widehat{t}_j$-paths have length $8m$. They can only be disconnected by deleting an edge with cost 1 in a type 1 tile in that column.

Let a solution S with total cost at most $c = \sum_{i=1}^{m}(|A_i|-1)+r$ of the just constructed BSP instance with pairs $(\bar{s}_i, \bar{t}_i)$ and $(\widehat{s}_j, \widehat{t}_j)$ be given. Let I be the set of indices of the rows in which unit cost edges are deleted. To block all shortest paths of one $\bar{s}_i$-$\bar{t}_i$-pair, edges with total cost at least $|A_i|-1$ have to be deleted ($i = 1,\ldots,m$). By definition c exceeds the sum of these values by r. So there are at most r rows in which edges with total cost larger than $|A_i|-1$ (namely the unit cost edges in the type 1 cells) may be deleted. Thus, $|I| \leq r$.

To block the unique shortest $\widehat{s}_j$-$\widehat{t}_j$-path a unit cost edge in a type 1 tile of the jth column has to be deleted. Let i be the row of that tile. Then $i \in I$ and $j \in A_i$ follow. Since this is true for each $j = 1,\ldots,n$ we have $\bigcup_{i\in I} A_i = \{1,\ldots,n\}$ and I is a solution of the SET COVER instance.

Vice versa, given a solution I of the SET COVER instance with $|I| \leq r$ we get a solution of BSP with total cost at most c by deleting all unit cost edges in the ith row for all $i \in I$ and the edge incident to $\bar{s}_i$ in the ith row for all $i \notin I$.

So it is possible to increase $\operatorname{dist}(\bar{s}_i, \bar{t}_i)$ and $\operatorname{dist}(\widehat{s}_j, \widehat{t}_j)$ for all i and j by deleting edges of total cost c if and only if there is a feasible solution of the SET COVER problem with r subsets.

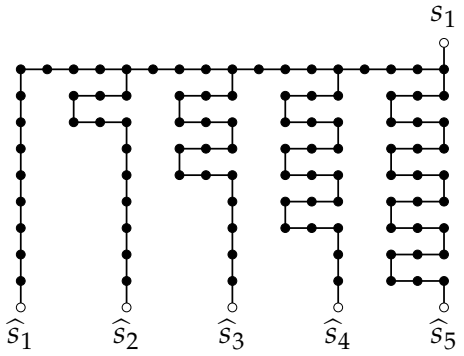

Figure 4.4. Vertices $\widehat{s}_j$ are connected to s_1 by paths of the same length.

To get an instance of BSP with two pairs we add a vertex s_1 and connect it to the vertices $\widehat{s}_j$ as shown in Figure 4.4 for $n = 5$. The edges have weights $c + 1$ and it holds $\text{dist}(\widehat{s}_j, s_1) = \text{dist}(\widehat{s}_h, s_1)$ for all j and h. Then we introduce vertices t_1, s_2, and t_2 and connect them to the vertices $\widehat{t}_j$, $\overline{s}_i$, and $\overline{t}_i$, respectively, in a similar way. Note that this does not introduce new shortest $\overline{s}_i$-$\overline{t}_i$-paths or $\widehat{s}_j$-$\widehat{t}_j$-paths. It holds $\text{dist}(\widehat{s}_j, \widehat{t}_h) \geq \text{dist}(\widehat{s}_j, \widehat{t}_j) + 4$ if $j \neq h$ and $\text{dist}(\overline{s}_i, t_h) \geq \text{dist}(\overline{s}_i, \overline{t}_i) + 2$ if $i \neq h$. Thus, increasing $\text{dist}(s_1, t_1)$ and $\text{dist}(s_2, t_2)$ is equivalent to increasing $\text{dist}(\overline{s}_i, \overline{t}_i)$ and $\text{dist}(\widehat{s}_j, \widehat{t}_j)$ for all i and j.

Clearly, this construction can be done in polynomial time. ❑

The following variation of this result uses unit cost edges only but needs general bipartite planar graphs.

Corollary 4.1
BSP with $k = 2$ pairs (s_i, t_i) is NP-hard, even for planar bipartite graphs G with unit costs.

PROOF. Let an instance of SET COVER as in the former proof be given. We may assume $|A_i| \geq 2$ since augmenting each set with two further elements (e.g., $n + 1$ and $n + 2$) will not change the feasibility of solutions. Then we use the same reduction as in the proof of Theorem 4.2. In that instance of BSP edge costs were integral and in the range $1, \ldots, c + 1$. So we can replace each edge e by a complete bipartite graph K_{2,c_e} with unit cost edges as shown in Figure 4.5 for $c_e = 6$. The size of the graph

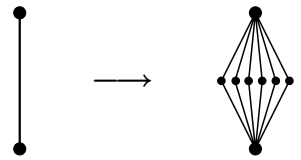

Figure 4.5. An edge with cost 6 is replaced by a $K_{2,6}$ with unit cost edges.

is still bounded polynomially by the size of the original SET COVER instance, since $c = \sum(|A_i| - 1) + r = \mathcal{O}(nm)$. This replacement doubles all distances in the graph, so the sets of shortest paths are unchanged. Deleting an edge with cost c_e in the original graph corresponds exactly to deleting a minimal cut from the K_{2,c_e}. Thus, there is a solution of BSP in the modified graph with total cost c' if and only if there is a solution of BSP in the grid graph with total cost c'. The just constructed graph is

planar and since lengths of cycles are doubled as well, it does not contain odd cycles and is therefore bipartite. More precisely, it contains only cycles whose lengths are a multiple of 4 since the original graph is already bipartite. ❑

With some other slight modification we can strengthen the result of Theorem 4.2.

Corollary 4.2
BSP with $k = 2$ pairs (s_i, t_i) is NP-hard, even for grid graphs G with maximum degree 3.

PROOF. The only vertices of degree 4 in the graph constructed in the proof of Theorem 4.2 are the two in each tile. To avoid them we first replace each edge e by a path with three edges with costs $c+1$, c_e, $c+1$. This triples the lengths of all paths and therefore does not change shortest paths, feasible solutions, and objective values in principle. Then the vertices with degree 4 can be replaced by an 8-cycle whose edges all have costs $c+1$ as shown in Figure 4.6. Again, this

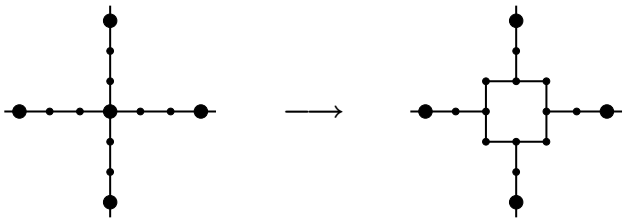

Figure 4.6. Vertices with degree 4 are replaced by 8-cycles. The large vertices are those of the graph constructed in the proof of Theorem 4.2.

preserves the optimal objective value and it is also applicable in type 1 tiles where vertices of degree 4 are adjacent. ❑

4.3 Triangle instances

So far we have seen that star instances can be solved in polynomial time while BSP is *NP*-complete if there are two disjoint pairs. The only configuration that is none of the above has pairs (r,s), (s,t), and (r,t). Corresponding instances are called *triangle instances*. In this section some remarks on this problem are given, but unfortunately its complexity will not be determined.

We can compute a feasible solution of a triangle instance by firstly blocking all shortest r-s-paths and then blocking all considered shortest paths starting at t in polynomial time as described in the proof of Theorem 4.1. Although the solutions of the two subproblems are optimal their union is not optimal generally for triangle instances, as the instance in Figure 4.7 shows. For this example with $\varepsilon > 0$ the unique

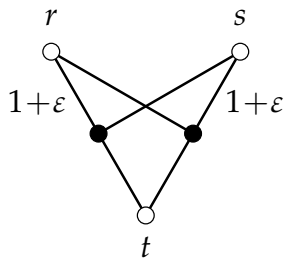

Figure 4.7. Triangle instance, all edge costs that are not given are unit costs.

optimal solution of the first subproblem is formed by the two unit cost edges incident to r or s, and the one for the second subproblem is made up by the two unit cost edges incident to t. For small ε the unique optimal solution of the triangle instance consists of the two edges with costs $1+\varepsilon$, yielding an approximation ratio of 2 for $\varepsilon \to 0$ which can be shown to be as bad as possible, see Theorem 7.1.

Dahlhaus et al. showed in [16] that MINIMUM MULTIWAY CUT, see definition on page 17, with three terminal vertices is *NP*-complete. Since this problem is closely related to triangle instances one could hope that their proof can be adopted to BSP. They give a counterexample showing that a certain approach does not lead to optimal solutions and use this instance to prove *NP*-hardness. So we take a look at a similar approach for BSP triangle instances.

For a set $X \subseteq V \setminus \{r,s,t\}$ let S_r be the set of edges $\{u,v\}$ of shortest r-s-paths and shortest r-t-paths satisfying $u \in X \cup \{r\}$, $v \notin X \cup \{r\}$, and $\operatorname{dist}(r,u) < \operatorname{dist}(r,v)$. Clearly, any shortest r-s-paths and any shortest r-t-paths contain an edge of S_r. Now set $c_e = 0$ for all $e \in S_r$ and determine the optimal solution S_{st} of the BSP instance with the pair (s,t). Then $S = S_r \cup S_{st}$ is a feasible solution of the triangle instance and there exists an X such that this solution is optimal. So if $f(X)$ is defined as the total cost of S then the minimum of the function f on the subsets of $V \setminus \{r,s,t\}$ corresponds to an optimal solution of the BSP instance. Note that f can be determined in polynomial time.

There is a result by Grötschel, Lovasz, and Schrijver on *submodular*

functions that is, functions f defined on the subsets of some finite set U satisfying

$$f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y) \tag{4.1}$$

for all $X, Y \subseteq U$. They showed in [27] that a submodular function can be minimized in polynomial time if it can be computed in polynomial time. So if the function f as defined above was submodular then this would lead to a polynomial algorithm for BSP. However, as the instance in Figure 4.8 shows, f is not submodular. For $X = \{x, z\}$ and $Y = \{y, z\}$

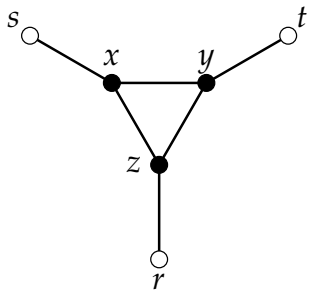

Figure 4.8. Triangle instance proving that f is not unimodular. All edges have unit costs.

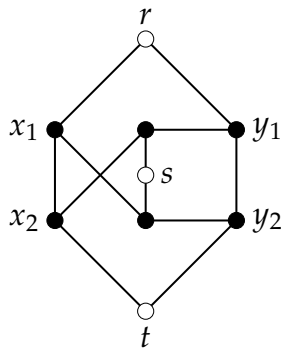

Figure 4.9. Triangle instance proving that f' is not unimodular. All edges have unit costs.

it holds $f(X) = f(Y) = 2$, $f(X \cap Y) = 3$, and $f(X \cup Y) = 2$ violating (4.1).

Similar to f one could define a function f' where only shortest r-t-paths are blocked in dependency of X and then an optimal solution of the star instance with pairs (s, r) and (s, t) is determined. The instance in Figure 4.9 proves that f' is not submodular, too. To see this consider the sets $X = \{x_1, x_2\}$ and $Y = \{y_1, y_2\}$.

For the MINIMUM MULTIWAY CUT Dahlhaus et al. also give a counterexample proving that a similar function for that problem is not submodular. Then they used the graph of the counterexample for their proof of *NP*-completeness. Unfortunately, their approach cannot be adopted for BSP since disconnecting *all* s-t-paths is indispensable for their construction. Therefore the complexity of these special BSP instances remains open.

4.4 Other special cases

The next theorem shows two more special cases of BSP that can be solved in polynomial time.

Theorem 4.3
BSP with a fixed number k of pairs and unit costs can be solved in polynomial time in outerplanar graphs and in graphs with bounded maximum degree.

PROOF. Let an outerplanar graph $G = (V, E)$ be given and consider a pair (s, t) of its vertices. We can assume that the vertices of G are the numbers $1, 2, \ldots, n$, arranged clockwise on a cycle in an outerplanar drawing and that $s = 1$. As mentioned in the introduction we may assume $\{s, t\} \notin E$. At least one of the vertices

$$\begin{aligned} u &= \max\{i \mid 2 \leq i \leq t-1,\ \{s,i\} \in E\}, \\ v &= \min\{i \mid t+1 \leq i \leq n,\ \{s,i\} \in E\} \end{aligned}$$

exists if the graph is connected. If only one of them exists, arguments similar to the following hold. Since there is no edge connecting a vertex $1, \ldots, u-1$ or $v+1, \ldots, n$ to a vertex $u+1, \ldots, v-1$ every s-t-path contains the vertex u or the vertex v (see the example in Figure 4.10). Thus, any shortest s-t-path uses the edge $\{s, u\}$ or the edge $\{s, v\}$.

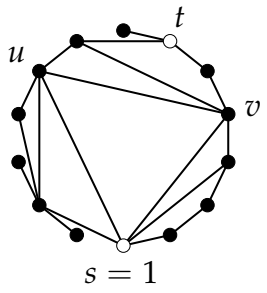

Figure 4.10. If $\{s, t\} \notin E$ each shortest s-t-path uses one of the edges $\{s, u\}$ and $\{s, v\}$.

Therefore deleting these edges blocks all shortest s-t-paths. This is true for all pairs, so there is a feasible solution of BSP with at most $2k$ edges. To find an optimal solution we only have to test all sets of at most $2k-1$ edges for feasibility, which is possible in polynomial time since k is fixed.

Likewise, for each graph G whose maximum degree is bounded from above by some $\Delta_{\max}$ there is a feasible solution of BSP with at most

$k\Delta_{\max}$ edges. It consists of all edges being adjacent to one of the vertices s_i. So in this case we can test all sets of up to $k\Delta_{\max} - 1$ edges in polynomial time to determine an optimal solution. ❑

Clearly, the proof of Theorem 4.3 leads to a polynomial time algorithm but its complexity is impractical. The question whether the structure of these special instances can be used to find a sophisticated algorithm remains open. Only for the case $\Delta(G) = 2$ we present a fast algorithm that even works if the number of pairs is part of the input data and costs are arbitrary.

Theorem 4.4
BSP in graphs G on n vertices with maximum degree $\Delta(G) = 2$ can be solved in $\mathcal{O}(n^3)$ time.

PROOF. If $\Delta(G) = 2$, components of G are paths or cycles. These components can be identified in $\mathcal{O}(nm) = \mathcal{O}(n^2)$ time using multiple breadth-first-searches. Then each subproblem can be solved separately and therefore we may assume that G is connected. We first consider the case where G is a path. If the BSP instance contains two pairs (s_i, t_i) and (s_j, t_j) such that s_i and t_i both are vertices of the unique shortest s_j-t_j-path then the second pair can be left out since s_j and t_j are disconnected once s_i and t_i are disconnected. To remove some of these redundant pairs we can use vertex labels $\ell(v)$ increasing along the path. With appropriate reordering we can assume $\ell(s_i) < \ell(t_i)$ for all $i = 1, \ldots, k$. Then we sort the pairs by increasing $\ell(s_i)$. If two pairs have the same $\ell(s_i)$ label then we omit the pair with larger $\ell(t_i)$ value. This can all be done in $\mathcal{O}(m + k \log k)$ which is smaller than $\mathcal{O}(n^3)$. After this preprocessing we have $k \leq n$ and $\ell(s_1) < \ell(s_2) < \ldots < \ell(s_k)$ for redefined pairs (s_i, t_i).

Now we consider a BSP instance with the same graph and with pairs $(s_i, t_i), \ldots, (s_k, t_k)$. For its optimal objective value $r(i)$ it holds

$$r(i) = \min \Big\{ c\{u,v\} + r(h) \;\Big|\; \begin{array}{l} \ell(s_i) \leq \ell(u) < \ell(v) \leq \ell(t_i), \\ h = \min\{j \mid \ell(s_j) \geq \ell(v)\} \end{array} \Big\}$$

where the minimum of the empty set is ∞, and $r(\infty) = 0$. Roughly speaking we remove an edge $\{u, v\}$ of the s_i-t_i-path and solve the BSP instance with those pairs of $(s_i, t_i), \ldots, (s_k, t_k)$ that are still connected. This equation leads to a *dynamic programming* approach. To compute in advance the minimum h for each edge $\{u, v\}$ as given in the formula

takes time $\mathcal{O}(n \log k)$. Then we can compute all values $r(i)$ in $\mathcal{O}(n^2)$ and thereby determine an optimal solution of the BSP instance.

Now we consider the case that the graph is a cycle. One of the edges of a (possibly non-unique) shortest s_1-t_1-path P has to be deleted. Once this edge is removed, an instance of BSP whose graph is a path remains. It can be solved in $\mathcal{O}(n^2)$ as described above. Testing each edge of P and keeping the best solution leads to an $\mathcal{O}(n^3)$ algorithm. ❑

BSP instances with unit costs and $\Delta(G) \leq 2$ can be solved even faster. To see this let G be a path and consider the open intervals $(\ell(s_i), \ell(t_i)) = \{x \in \mathbb{R} \mid \ell(s_i) < x < \ell(t_i)\}$ defined using the vertex labels $\ell(v)$, see Figure 4.11. It is easy to see that the BSP instance is

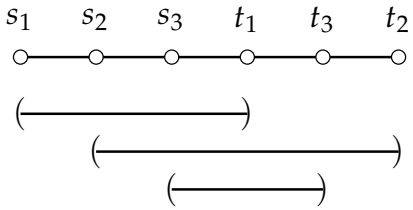

Figure 4.11. Intervals $(1, 4)$, $(2, 6)$, and $(3, 5)$ corresponding to the three pairs of the BSP instance.

equivalent to solving MINIMUM CLIQUE COVER in the interval graph H on these intervals, that is, finding a minimum set of complete subgraphs of H such that each vertex of H is contained in at least one of them. If G is a cycle then one can construct a circular arc graph in a similar way. For these cases MINIMUM CLIQUE COVER can be solved in $\mathcal{O}(k \log k)$ and $\mathcal{O}(k^2)$ where k is the number of intervals and segments, respectively, see [29]. So if $k \leq n$ (which can always be achieved by preprocessing) we get an $\mathcal{O}(n^2)$ algorithm for unit cost instances.

As long as all components are paths, BSP is equivalent to MULTI CUT since each path is a shortest path. Trivially, this is also true if the graph is a tree. Therefore the following result proved by Garg, Vazirani, and Yannakakis in [25] for MULTI CUT is true for BSP as well. Because of its simplicity we also review the proof.

Theorem 4.5 (Garg, Vazirani, and Yannakakis)
BSP is NP-hard and APX-hard, even in stars $K_{1,n}$ with unit costs.

PROOF. The proof is by reduction from MINIMUM VERTEX COVER. Given a graph $G = (V, E)$ as an instance of this problem, we consider the unit cost BSP instance with vertex set $V \cup \{w\}$, edges $\{v, w\}$ for all

$v \in V$, and pairs (s,t) for all $\{s,t\} \in E$. Then for each edge $\{u,v\}$ of G one has to delete $\{u,w\}$ or $\{v,w\}$ in the BSP instance corresponding to selecting u or v for the vertex cover. Thus, this reduction preserves objective values, and consequently, unless $P = NP$, BSP is not approximable within $7/6$ like MINIMUM VERTEX COVER, see [32]. ❑

4.5 Summary

As a conclusion Table 4.1 summarizes the results obtained in this chapter. Several special cases were discussed, but there are still open problems.

number of pairs	costs	graph properties	status	reference
1	—	—	*P*	Th 4.1
2	—	grid graph, $\Delta = 3$	*NP*	Co 4.2
2	unit	planar and bipartite	*NP*	Co 4.1
fixed	unit	bounded degree	*P*	Th 4.3
fixed	unit	outerplanar	*P*	Th 4.3
arbitrary	unit	stars $K_{1,n}$	*NP*	Th 4.5
arbitrary	—	$\Delta = 2$	*P*	Th 4.4

Table 4.1. Overview of complexity results.

An interesting question besides complexity of special cases is approximability of the *NP*-hard cases. Theorem 4.5 states that BSP with an arbitrary number of pairs is *APX*-hard, that is, there is no polynomial time approximation scheme (*PTAS*) as long as $P \neq NP$, even if the graph has a simple structure and edges have unit costs. If the number k of pairs is fixed then it is not clear whether or not a *PTAS* exists. In Chapter 7, a constant factor approximation algorithm for instances with fixed k is presented, but if k is not fixed then such a result is missing. Therefore in both cases the analysis of the approximability is not tight.

CHAPTER 5

Results in Extremal Graph Theory

THERE are several questions in extremal graph theory concerned with BSP. Some of them are useful for studies of complexity and approximation, while some others are interesting in their own right.

5.1 Unit cost instances with large optimal solutions

In this section we ask how large the optimal objective value of a BSP instance can be. More precisely, we seek instances having unit costs and a given number of vertices and pairs such that the optimal objective value is maximal.

Theorem 5.1
For instances of BSP with unit costs, n vertices, and k disjoint vertex pairs (s_i, t_i) the optimal objective value is at most

$$\begin{array}{ll} k(n-k-1) & \text{if } n \geq 2k+1, \\ \frac{1}{4}n^2 - \frac{1}{2}n + 1 & \text{if } n = 2k. \end{array}$$

Both bounds are tight.

PROOF. Consider a graph G with n vertices and k disjoint vertex pairs (s_i, t_i) of G. Let k_1 be the number of pairs with distance 1. We can assume that (s_i, t_i) with $i = 1, \ldots, k_1$ are those pairs. Now all considered

distances can be increased by removing the edges $\{s_i, t_i\}$ for $i = 1, \ldots, k_1$ and the edges of the cut $S = \delta_G(\{s_{k_1+1}, \ldots, s_k\})$, that is, all edges being incident to exactly one of the vertices $s_{k_1+1}, \ldots, s_k$. Since $\{s_i, t_i\} \notin S$ for $i = k_1 + 1, \ldots, k$, we have

$$|S| \leq (k - k_1)(n - (k - k_1)) - (k - k_1).$$

Therefore the total number of edges to remove is

$$\begin{aligned} k_1 + |S| &\leq k_1 + (k - k_1)(n - (k - k_1)) - (k - k_1) \\ &= k + \tfrac{1}{4}n^2 - n + 1 - \left(k_1 + \tfrac{1}{2}n - k - 1\right)^2. \end{aligned} \tag{5.1}$$

Consider the case $n \geq 2k + 1$. Then the right hand side of (5.1) is minimal if $k_1 = 0$ and for this value we get $k_1 + |S| \leq k(n - k) - k$, which proves the first assertion.

In the case $n = 2k$ the expression in (5.1) is minimal if $k_1 = 1$, resulting in the claimed bound $\frac{1}{4}n^2 - \frac{1}{2}n + 1$.

To prove the tightness of the bounds we consider a graph G with vertices $s_1, s_2, \ldots, s_k$, $t_1, t_2, \ldots, t_k$, and $v_1, v_2, \ldots, v_{n-2k}$ and all edges except for $\{s_i, t_i\}$, $i = 1, \ldots, k$ (see Figure 5.1 for $k = 3$ and $n = 8$). Whether or not edges $\{v_i, v_j\}$ are present is non-relevant since they are not part of any of the shortest s_i-t_i-paths.

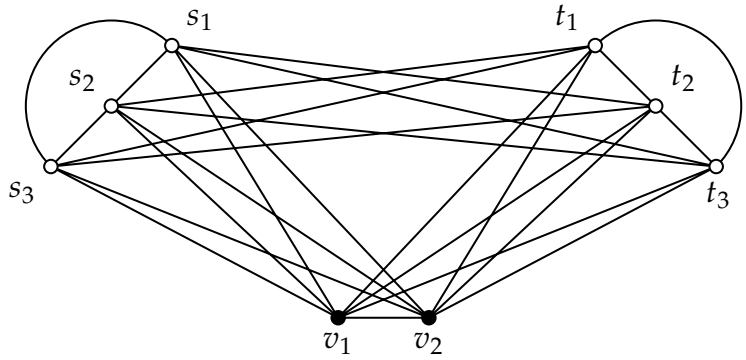

Figure 5.1. A unit cost BSP instance with $n = 8$ vertices, $k = 3$ pairs, and optimal objective value of 12.

We consider the following shortest paths only: For $i = 1, \ldots, k$ and $j = 1, \ldots, n - 2k$ we have the paths (s_i, v_j, t_i), their number is $k(n - 2k)$. For $1 \leq i < j \leq k$ we have the paths (s_i, t_j, t_i) and (s_i, s_j, t_i). The number of these paths is $\binom{k}{2}$ for both types. All these paths are edge-disjoint

and their total number is

$$k(n-2k) \quad + \quad 2\binom{k}{2} = k(n-k-1).$$

Since one edge of each path has to be deleted, this proves the assertion for $n \geq 2k+1$.

For $n = 2k$ we add the edge $\{s_k, t_k\}$ to the just described graph. Then we have the shortest paths (s_i, t_j, t_i) and (s_i, s_j, t_i) with $1 \leq i < j \leq k$ and (s_k, t_k). Again, they are all disjoint and their number is

$$2\binom{k}{2} + 1 = \frac{1}{4}n^2 - \frac{1}{2}n + 1,$$

thus completing the proof. ❑

5.2 The duality gap

As mentioned in the introduction the dual of the LP-relaxation of a MIP for BSP with unit costs has the following combinatorial interpretation: For a given BSP instance and the set $\mathcal{P}$ of considered shortest paths, find a maximum subset of $\mathcal{P}$ containing edge-disjoint paths only. We will call this problem PACKING SHORTEST PATHS (PSP). Figure 1.2 in the introduction shows an instance where optimal objective values of BSP and PSP are different. This leads to the question how large the ratio of the optimal objective values of the two problems can be.

Theorem 5.2
For any BSP instance with k pairs it holds

$$\frac{z_{\text{BSP}}}{z_{\text{PSP}}} \leq k,$$

where z_{BSP} is the optimal objective value of the BSP instance and z_{PSP} is the optimal objective value of the corresponding PSP instance. This bound is best possible, even for grid graphs.

PROOF. For instances with $k = 1$ pair, we have already seen in Theorem 4.1 on page 30 that any minimum s-t-cut in the shortest paths digraph D_s is an optimal solution of BSP. Since directed s-t-paths in D_s correspond exactly to shortest s-t-paths in G the well-known result by Ford and Fulkerson (see [21]) can be applied, proving that BSP and PSP have the same objective value.

For instances with $k \geq 2$ pairs let z_i be the optimal objective value of the BSP instance with one pair (s_i, t_i). Since the union of the corresponding optimal solutions is feasible for the original instance it holds

$$z_{\text{BSP}} \leq \sum_{i=1}^{k} z_i.$$

By the pigeonhole principle there is an index i such that $z_{\text{BSP}} \leq k z_i$. As mentioned in the previous paragraph there are z_i edge-disjoint shortest s_i-t_i-paths, so $z_i \leq z_{\text{PSP}}$ and this proves the asserted inequality.

To prove that this bound is best possible we consider BSP instances with grid graphs constructed as follows. The vertices $s_1, s_2, \ldots, s_k$ are arranged from top to bottom in a column on the left side, the vertices $t_1, t_2, \ldots, t_k$ are arranged from bottom to top in a column on the right side. For the first pair we insert two paths being parallel horizontal lines. For the ith pair we insert two paths starting at s_i going rightwards, then going alternatingly upwards and rightwards crossing all paths inserted before, and finally going rightwards to t_i, see Figure 5.2 and the title page of this work for $k = 3$.

Because of the arrangement any s_i-t_i-path and any s_j-t_j-path with $i \neq j$ have a crossing. Since the graph is planar a crossing is a common interior vertex of the two paths. Since the maximum degree is 3 the two paths share an edge incident to such a vertex. Clearly, any two s_i-t_i-paths share the edge adjacent to t_i. Thus, no two edge-disjoint shortest paths exist, that is, $z_{\text{PSP}} = 1$.

The $2k$ paths used to define this graph are shortest paths since they use edges going upwards and rightwards only. By construction no edge is used by three of those paths, so k edges have to be deleted to block all of them. Thus, $z_{\text{PSP}} \geq k$ which implies $z_{\text{BSP}}/z_{\text{PSP}} \geq k$. ❑

5.3 The maximal number of shortest paths

It is well-known that the number of shortest paths in graphs may increase exponentially with the number of vertices. Besides the asymptotics one may ask for a tight bound on the number of shortest s-t-paths in a graph on n vertices. We will use the bound determined in this section to prove an approximation guarantee in Chapter 7.

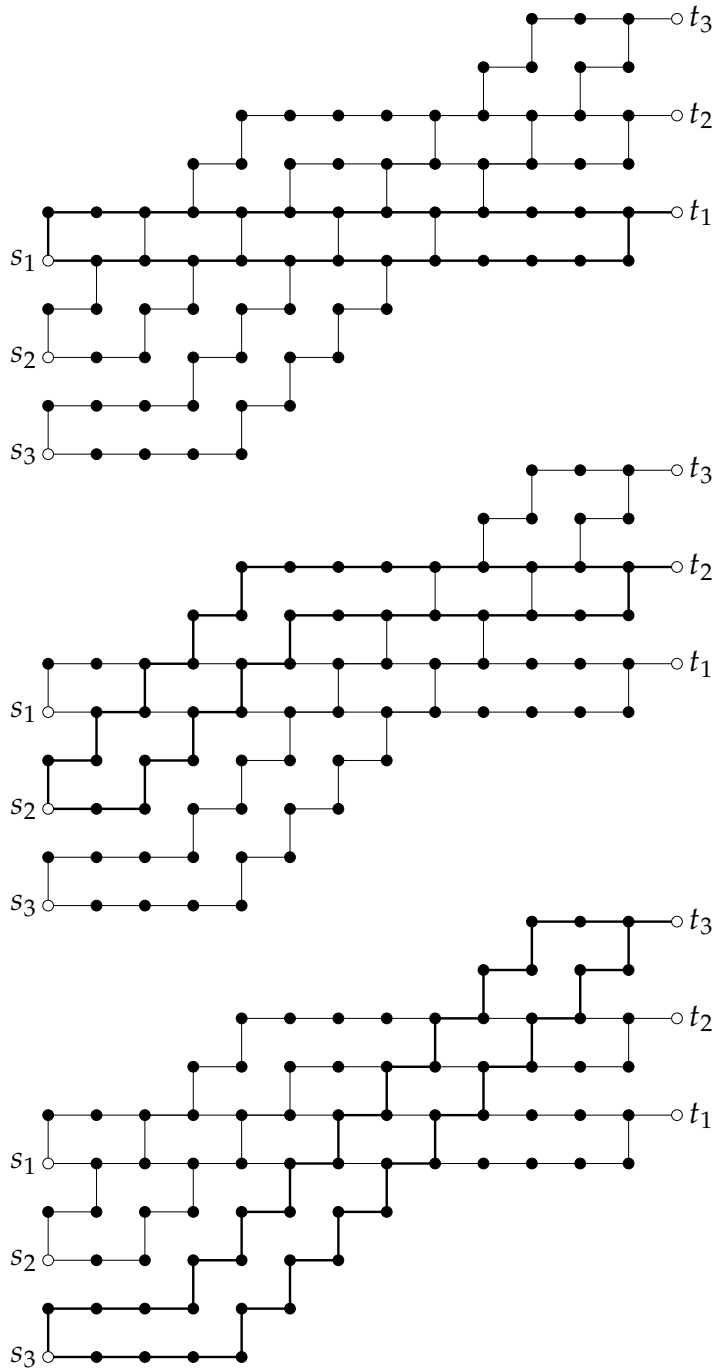

Figure 5.2. A BSP instance being the union of two s_i-t_i-paths for each $i = 1, 2, 3$.

Theorem 5.3
Given a graph or digraph $G = (V, E)$ on n vertices and $s, t \in V$, there are at most

$$4 \cdot 3^{(n-6)/3}, \quad 2 \cdot 3^{(n-4)/3}, \quad 3^{(n-2)/3}$$

shortest s-t-paths for $n \equiv 0, 1, 2 \pmod 3$, respectively. This bound is tight, it is attained for graphs as shown in Figure 5.3.

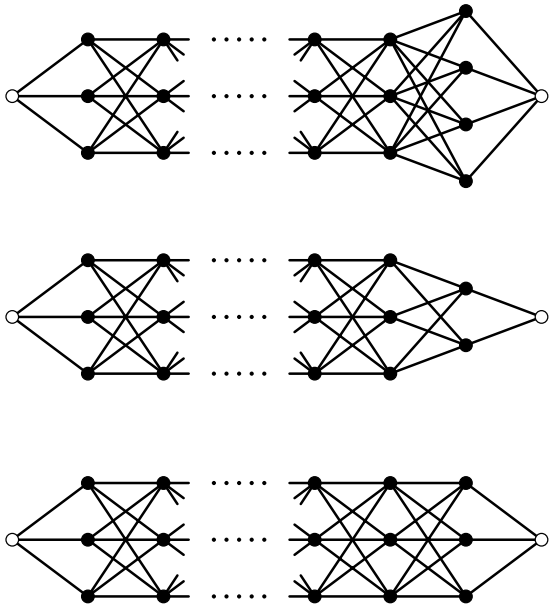

Figure 5.3. Undirected graphs with maximum number of shortest paths between white vertices and $n \equiv 0, 1, 2 \pmod 3$ vertices. In the corresponding digraphs edges are oriented from left to right.

PROOF. For a given graph or digraph $G = (V, E)$ and vertices $s, t \in V$, let $d = \operatorname{dist}(s, t) - 1$ be the number of interior vertices on shortest s-t-paths and let n_i ($i = 1, \ldots, d$) be the number of vertices v with $\operatorname{dist}(s, v) = i$. Since any shortest s-t-path uses exactly one vertex of distance i for each $i = 1, \ldots, d$ there are at most $\prod n_i$ such paths. This bound is tight if any two vertices u and v with $\operatorname{dist}(s, u) = \operatorname{dist}(s, v) + 1$ are adjacent. So the maximal number of shortest s-t-paths in a graph with n vertices is given by the system

$$\max \prod_{i=1}^{d} n_i \quad \text{s. t.} \quad \sum_{i=1}^{d} n_i = n - 2, \quad n_i, d \in \mathbb{N}. \tag{5.2}$$

In other words, we seek a finite sequence (n_i) of positive integers such that the sum of its elements is fixed and such that the product of its elements is maximal. Let an optimal sequence (n_i) be given and let d be the number of its elements. Assume that there is an index i with $n_i \geq 5$. Since $2(n_i - 2) = 2n_i - 4 > n_i$ we could increase the product by replacing n_i with $n_i - 2$ and expanding the sequence by adding an element $n_{d+1} = 2$. So $n_i \leq 4$ holds for all $i = 1, \ldots, d$.

Analogously, the subsequences $(1, x)$, $(2, 4)$, $(4, 4)$, and $(2, 2, 2)$ of (n_i) can be replaced by $(1 + x)$, $(3, 3)$, $(3, 3, 2)$, and $(3, 3)$, respectively, yielding a feasible solution of system (5.2) with increased objective value. Therefore in an optimal solution the number 2 occurs at most twice and the number 4 occurs at most once, all other elements are 3. There is no optimal solution containing both, 2 and 4.

Since interchanging the subsequences $(2, 2)$ and (4) does not affect the objective function, optimal solutions are $n_i = 3$ for $i = 1, \ldots, d - 1$ and

$$\begin{aligned} n_d &= 3, \quad d = (n-2)/3 && \text{for } n \equiv 2 \pmod 3, \\ n_d &= 2, \quad d = 1 + (n-4)/3 && \text{for } n \equiv 1 \pmod 3, \\ n_d &= 4, \quad d = 1 + (n-6)/3 && \text{for } n \equiv 0 \pmod 3, \end{aligned}$$

thus proving the assertion. ❑

CHAPTER 6

Optimal Solution Approaches

FOR problems that are *NP*-hard as shown for BSP in Chapter 4 it is unlikely that an exact solution can be determined in polynomial time. Nevertheless, one may ask for approaches to solve small instances of a problem in acceptable time. A common technique to achieve this is to formulate the problem as a mixed integer program (MIP) and to solve it using standard algorithms or specialized variations of them. In this chapter we discuss a straightforward MIP formulation and a more elaborated one.

6.1 Set cover MIP formulation

The most obvious MIP formulation was already mentioned in the introduction, primarily to present its dual problem and its combinatorial interpretation. Thinking of BSP as a special SET COVER PROBLEM where shortest paths are the elements to be covered and edges correspond to the subsets one may select, leads to the MIP

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e \\ \text{s.t.} \quad & \sum_{e \in P} x_e \geq 1 \quad \text{for all } P \in \mathcal{P} \\ & x_e \in \{0,1\} \quad \text{for all } e \in E, \end{aligned} \tag{6.1}$$

where c_e are the edge costs and $\mathcal{P}$ is the set of considered shortest paths. Beneath the fact that solving MIPs is *NP*-hard, this special MIP may have an exponential number of inequalities, see Theorem 5.3

on page 44. Anyhow, given some infeasible x we can determine a violated inequality by removing all edges e with $x_e = 1$ and then determining $\text{dist}_{\widetilde{G}}(s_i, t_i)$ for all $i = 1, \ldots, k$ in the reduced graph $\widetilde{G}$. If $\text{dist}_{\widetilde{G}}(s_i, t_i) = \text{dist}_G(s_i, t_i)$ for the original graph G and some index i then a corresponding shortest s_i-t_i-path in $\widetilde{G}$ specifies an unsatisfied inequality. Otherwise it holds $\text{dist}_{\widetilde{G}}(s_i, t_i) \geq \text{dist}_G(s_i, t_i) + 1$ for all $i = 1, \ldots, k$ and the solution is feasible. Therefore, we can determine an inequality that is not fulfilled or verify feasibility in $\mathcal{O}(km)$ time using k breadth-first-searches.

Analogously, we can determine violated inequalities for the relaxation, that is, the linear program (LP) identical to the MIP in (6.1) but with $x_e \in [0, 1]$ instead of $x_e \in \{0, 1\}$. For this purpose we determine shortest s_i-t_i-paths in G with respect to the edge lengths $1 + x_e$. If there is an s_i-t_i-path with total length smaller than $\text{dist}_G(s_i, t_i) + 1$ then it has $\text{dist}_G(s_i, t_i)$ edges and it is one of the paths to block in the BSP instance. Moreover, the sum of the x-values of its edges is smaller than 1 and the corresponding inequality is not fulfilled. Otherwise the length of the shortest s_i-t_i-path with respect to the edge lengths $1 + x_e$ is at least $\text{dist}_G(s_i, t_i) + 1$. Then the sum of its x-values is not smaller than 1 or it has at least $\text{dist}_G(s_i, t_i) + 1$ edges. In both cases inequalities corresponding to shortest s_i-t_i-paths of the BSP instance are satisfied. So for the relaxation we can verify the feasibility of a given x or determine a violated inequality in $\mathcal{O}(kn^2)$ using the basic Dijkstra algorithm.

With the just described approach we can solve the relaxation in polynomial time using ellipsoid method. Furthermore, we can find optimal solutions of BSP using the following *branch-and-cut* approach. We start with a list of open problems, the so-called branch-and-cut nodes, containing the MIP (6.1) only. Then we iteratively take an open problem out of the list and determine the optimal solution of its relaxation. We select some non-integral variable x_e and add the reduced problems with $x_e = 0$ and $x_e = 1$ to the list of open problems. If all variables are integral then we found a feasible solution of the BSP instance and we save the best of such solutions. If the optimal objective value of the relaxation is not smaller then the total cost of the best integral solution we have seen so far, then we do not add the two reduced problems to the list since they can not lead to better solutions.

This standard technique leads to a finite algorithm, but time and memory consumption both may be exponential in the number of vertices of the BSP instance.

6.2 Flow based MIP formulation

The main disadvantage of the MIP presented in the previous section is the possibly exponential number of inequalities. In this section we derive an equivalent MIP formulation which has only a polynomial number of inequalities. It is based on the LP for MINIMUM CUT which in turn is the dual of the LP of MAXIMUM FLOW. These LPs both have integral optimal solutions, but for BSP we have to force variables to be integers.

In addition to the edge variables x_e we have vertex variables y_v^i for each vertex $v \in V$ and each $i = 1, \ldots, k$. They represent some kind of distance between s_i and v. The MIP is

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e & & \\ \text{s. t.} \quad & x_{\{u,v\}} + y_u^i - y_v^i \geq 0 & & \text{for all } (u,v) \in D_{s_i} \\ & & & \text{and all } i = 1, \ldots, k \\ & y_{s_i}^i = 0 & & \text{for all } i = 1, \ldots, k \\ & y_{t_i}^i = 1 & & \text{for all } i = 1, \ldots, k \\ & x_e \in \{0,1\} & & \text{for all } e \in E, \end{aligned} \tag{6.2}$$

where D_{s_i} is the shortest path digraph with root s_i, see Definition 4.1 on page 29. This MIP consists of one MINIMUM CUT formulation for each pair in the corresponding shortest paths digraph. Each of these formulations has its own vertex variables y_v^i but uses common edge variables x_e. The MIPs in (6.1) and in (6.2) are some kind of equivalent as the following lemma shows.

Lemma 6.1
Let P_1 and P_2 be the polytopes of the relaxations of the MIPs in (6.1) and in (6.2), respectively, with $x_e \in [0,1]$ instead of $x_e \in \{0,1\}$. Then P_1 is the projection of P_2 into the space of the edge variables x_e.

PROOF. Consider a point $(x,y) \in P_2$ and an inequality of the MIP in (6.1). Let $(v_1, v_2, \ldots, v_d)$ with $v_1 = s_i$ and $v_d = t_i$ be the shortest s_i-t_i-path corresponding to this inequality. Then (v_j, v_{j+1}) is an edge of D_{s_i} for $j = 1, \ldots, d-1$. If we add the corresponding inequalities in (6.2) we get

$$0 \leq \sum_{j=1}^{d-1} \left(x_{\{v_j,v_{j+1}\}} + y_{v_j}^i - y_{v_{j+1}}^i \right) = \sum_{j=1}^{d-1} x_{\{v_j,v_{j+1}\}} \quad + y_{s_i}^i - y_{t_i}^i$$

which is equivalent to

$$\sum_{j=1}^{d-1} x_{\{v_j,v_{j+1}\}} \geq 1.$$

Therefore x satisfies all inequalities in (6.1) and the projection of P_2 is a subset of P_1.

Now consider a point $x \in P_1$. Let y_v^i be the length of a shortest s_i-v-paths in D_{s_i} with respect to the edge lengths x_e. Then the triangle inequalities $y_u^i + x_{\{u,v\}} \geq y_v^i$ are true for all edges in D_{s_i} and therefore $(x, y) \in P_2$ holds and P_1 is a subset of the projection of P_2, completing the proof. ❑

So the MIP in (6.2) can be used to solve BSP. Another advantage of this MIP is that some edge variables x_e will have integral values in an optimal solution of the relaxation if some other variables are fixed to be 1 or 0. The variables that have to be fixed are the following.

Definition 6.1
An edge e of a BSP instance is called critical *if there are two indices i and j such that e is an edge of a shortest s_i-t_i-path and an edge of a shortest s_j-t_j-path.*

If variables of critical edges are fixed individually to 1 or 0 then the reduced MIP consists of k independent MINIMUM CUT LPs each of which has an integral optimal solution. In a branch-and-bound approach to solve this MIP one would branch on variables of critical edges only. Since their number is typically rather small, the number of necessary branches should be much smaller than in the branch-and-cut algorithm described in the previous section. Moreover, one may expect the effort for solving the MIP in (6.2) to increase with the number of critical edges. We will see in Chapter 8 whether this intuition holds for practical computations. In Section 7.3 we will discuss the relaxation of (6.2) again to derive an approximation algorithm.

CHAPTER 7

Approximation Algorithms

HARD optimization problems in practice can be faced in two different ways. Either one asks for optimal solutions and accepts possibly long computation times, or one asks for *almost optimal solutions* which can be found faster for some problems. This chapter discusses the second approach for BSP. We will consider four ways to approximate this problem that differ in running time, performance guarantee, and implementation effort.

7.1 Decomposition into easy subproblems

As we have already seen in Theorem 4.1 on page 30, it is possible to find exact solutions of BSP instances with one pair in polynomial time. So if we have an instance of BSP with more than one pair it is straightforward to solve BSP for each pair separately to get an approximation.

Algorithm 7.1 (Decomposition)
Given an instance of BSP, we determine minimum cost sets S_i increasing $\text{dist}(s_i, t_i)$ for $i = 1, \ldots, k$ as described in the proof of Theorem 4.1 and set $S = \bigcup_i S_i$.

Since deleting S_i from the graph blocks all shortest s_i-t_i-paths this is also true for S and therefore Algorithm 7.1 determines a feasible solution of the given instance of BSP in polynomial time. Furthermore, the following theorem holds.

Theorem 7.1
Algorithm 7.1 gives a factor k approximation for BSP*, where k is the number of pairs. This factor is best possible even for caterpillars with maximum degree* 3 *and unit costs.*

PROOF. Let an instance I of BSP with k pairs and optimal objective value C_{opt} be given. For $i = 1, \ldots, k$ denote by C_i the minimum cost of an edge set whose deletion disconnects all shortest s_i-t_i-paths. By definition any solution of the original problem blocks all such paths. Therefore it holds $C_i \leq C_{\text{opt}}$ for $i = 1, \ldots, k$ and consequently

$$\sum_{i=1}^{k} C_i \leq k C_{\text{opt}},$$

proving the asserted approximation guarantee.

Figure 7.1 shows a BSP instance whose graph is a grid graph and a caterpillar unit costs and k vertex pairs. Algorithm 7.1 may select

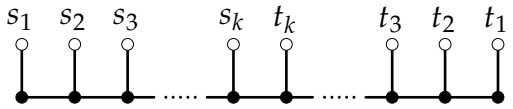

Figure 7.1. Bad instances for Algorithm 7.1.

one vertical edge for each pair, while the optimal solution is to delete the horizontal edge on the unique s_k-t_k-path. Thus, the ratio between the approximation and the optimal solution may be k. Note that the approximate solution described above is minimal, that is, excluding any edge from this set makes it infeasible. ❑

For the instance proving the lower bound the solution is not unique. We can assure this by changing the costs of the vertical edges to $1 + \varepsilon$ dropping the unit-cost property.

If the vertex pairs are not disjoint (for $k > n/2$ this is always the case) Algorithm 7.1 can be improved. If we interpret the pairs (s_i, t_i) as edges of a *demand graph* then this graph can be partitioned into $\overline{k} \leq k - 1$ stars. Therefore the given instance can be divided into at most $\overline{k}$ subproblems of the type introduced in Theorem 4.1, yielding a factor $\overline{k}$ approximation. Even if the demand graph is complete it can be partitioned into at most $n - 1$ stars where $n = |V|$. Thus, $\overline{k} \leq n - 1$ holds and the decomposition algorithm is also an $(n - 1)$-approximation.

7.2 Greedy approach

Our second approximation algorithm is motivated by the fact that BSP is a special SET COVER problem where the elements to be covered are shortest paths and the subsets to be selected are edges. Since there is the well-known greedy algorithm for SET COVER it is suggesting to apply it to BSP. To do this we have to iteratively delete a *most cost effective edge*, that is, an edge with maximal ratio between the number of currently unblocked shortest paths using this edge and its edge cost.

Definition 7.1
The number of shortest s-t-paths using an edge e is called shortest s-t-path frequency *of e and is denoted by $f_{s,t}(e)$.*

The just defined characteristic is similar to *stress centrality for edges*, see [47], except that for the latter not only some given pairs but all pairs are considered.

The straightforward way to find a most cost effective edge is to determine the values $f_{s_i,t_i}(e)$ for all $i = 1, \ldots, k$ and all $e \in E$. In fact, this is not necessary in each iteration, see the summary at the end of this chapter.

Lemma 7.1
Given a graph $G = (V, E)$ and vertices $s, t \in V$ the shortest s-t-path frequencies $f_{s,t}(e)$ can be determined in $\mathcal{O}(m)$ time for all $e \in E$, where $m = |E|$.

PROOF. Let $\{u, v\} \in E$ be an edge of a shortest s-t-path, that is

$$\operatorname{dist}(s, u) + \operatorname{dist}(v, t) + 1 = \operatorname{dist}(s, t).$$

It holds

$$f_{s,t}\big(\{u, v\}\big) = p_s(u)p_t(v),$$

where $p_s(u)$ and $p_t(v)$ are the numbers of shortest s-u-paths and shortest v-t-paths, respectively. So it suffices to determine $p_s(u)$ and $p_t(v)$ for all u and v. Let

$$P(u) = \{\overline{u} \mid dist(s, \overline{u}) + 1 = \operatorname{dist}(s, u), \ \ \{\overline{u}, u\} \in E\}$$

be the set of direct predecessors of u on shortest s-t-paths. Then the recursion

$$p_s(u) = \begin{cases} 1 & \text{for } \ u = s, \\ \displaystyle\sum_{u' \in P(u)} p_s(u') & \text{for } \ u \neq s \end{cases}$$

holds. The values $p_t(v)$ can be determined in a similar way. Each of the three parts breadth-first-searches, recursion, and forming products can be done in $\mathcal{O}(m)$ time. ❑

With this result we can apply the greedy algorithm for SET COVER.

Algorithm 7.2 (Greedy)
Given a graph $G = (V, E)$ with edge costs c_e and a list of vertex pairs (s_i, t_i) compute the shortest path frequencies $f_{s_i,t_i}(e)$ for each i and each $e \in E$. Then set $f = \sum_i f_{s_i,t_i}$ and delete an edge e with maximal ratio $f(e)/c_e$ from G. Remove each pair from the list whose distance is increased by this deletion. If the list is not empty then iterate.

Determining the values $p_{s_i}(\cdot)$ and $p_{t_i}(\cdot)$ for all pairs and thus determining $f(e)$ for all edges can be done in $\mathcal{O}(km)$ time. Since we delete at most m edges the total complexity is $\mathcal{O}(km^2)$. To get an approximation guarantee for this algorithm we can use the upper bound on the number of shortest paths determined in Theorem 5.1 on page 41 and apply the well-known log-result for the greedy algorithm for SET COVER, yielding the following approximation guarantee.

Theorem 7.2
For instances of BSP with n vertices and k pairs the greedy algorithm determines a factor $(\ln k + n \ln 3/3 + 1 - 2 \ln 3/3)$-approximation.

PROOF. Algorithm 7.2 applies the well-known greedy SET COVER approximation, where the shortest paths correspond to the elements to be covered and the edges correspond to the subsets. The greedy algorithm for SET COVER is a factor $1 + \ln h$ approximation where h is the number of elements to be covered . With Theorem 5.3 on page 44 we get

$$\max\{4 \cdot 3^{(n-6)/3},\ 2 \cdot 3^{(n-4)/3},\ 3^{(n-2)/3}\}k \ = \ 3^{(n-2)/3}k$$

as an upper bound on the number of shortest paths. Thus, the greedy algorithm has an approximation factor of

$$\ln\left(3^{(n-2)/3}k\right) + 1 \ = \ \ln k + \frac{\ln 3}{3}n + 1 - \frac{2\ln 3}{3}.$$

❑

This approximation factor is about $\ln k + 0.366n - 0.268$. The following theorem shows that this bound is *nearly* tight, at least for $k = 1$ pair.

Theorem 7.3
For instances of BSP *with n vertices and one pair the greedy algorithm may determine a solution that is $(0.331n - 0.158)$ times as expensive as an optimal solution.*

PROOF. The examples used to prove the assertion consist of $h + 1$ triples of vertices, where any two consecutive triples form a complete bipartite graph $K_{3,3}$. Furthermore, the vertices of the first triple are adjacent to a vertex s and the vertices of the last triple are adjacent to a vertex w, which in turn is adjacent to a vertex t, see Figure 7.2. The

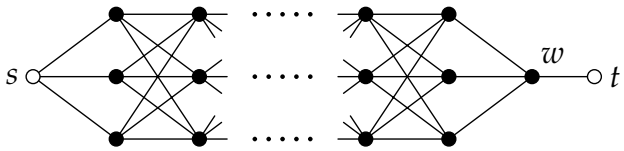

Figure 7.2. Bad instances for the greedy algorithm.

edge $\{w, t\}$ has cost 1, the costs of the edges of the h bipartite graphs are given in Figure 7.3. Only in the leftmost bipartite graph are costs

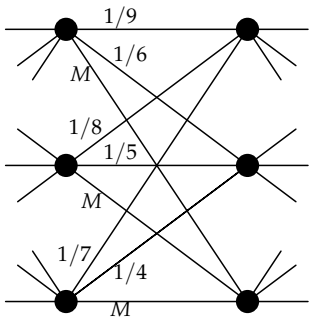

Figure 7.3. Edge costs of the instance in Figure 7.2.

M replaced by 1, $1/2$, and $1/3$ in arbitrary order. All remaining edges have costs M. Since all shortest s-t-paths use the edge $\{w, t\}$ it has a maximal path frequency and therefore edges with costs $M > 1$ are not selected by the greedy algorithm. We prove that all edges having costs smaller than M are deleted. The proof is by induction on the number h of bipartite graphs.

For $h = 1$ the shortest path frequency $f(e)$ of $e = \{w, t\}$ is equal to the number of edges in the bipartite graph during all iterations. The

edges of the bipartite graph have a frequency of 1. Before deleting the ith edge, $\{w,t\}$ has an effectiveness ratio of $10-i$ and the edge with cost $1/(10-i)$ has also a ratio of $10-i$. Therefore the latter one may be selected by the greedy algorithm. In other words, all edges of the bipartite graph may be deleted in order of increasing costs.

For $h \geq 2$ we prove that in the first six iterations all edges with costs smaller than M may be removed in order of increasing costs in the rightmost bipartite graph. As long as this is the case the following properties hold.

(i) All shortest s-t-paths using the edge $e = \{w,t\}$ can be constructed by selecting one edge of the rightmost bipartite graph and one vertex of each of the first $h-1$ triples. Thus, before deleting the ith edge the shortest path frequency is $f(e) = (10-i)3^{h-1}$ and it holds $f(e)/c_e \leq (10-i)3^{h-1}$.

(ii) For edges e of the rightmost bipartite graph all shortest s-t-paths using e can be constructed by selecting one vertex of each of the first $h-1$ triples. The shortest path frequency is $f(e) = 3^{h-1}$ in each iteration and right before deleting the ith edge it holds $f(e)/c_e = (10-i)3^{h-1}$ for the most cost effective edge.

(iii) For edges e of the first $h-2$ bipartite graphs all shortest s-t-paths using e can be constructed by selecting one vertex of each of the first $h-1$ triples that do not contain a vertex of e and one edge of the rightmost bipartite graph. Therefore, before deleting the ith edge the shortest path frequency is $f(e) = (10-i)3^{h-3}$ and it holds $f(e)/c_e \leq (10-i)3^{h-1}$.

So it remains to prove that in each of the first six iterations no edge of the $(h-1)$th bipartite graph is more cost effective than the cheapest edge of the rightmost bipartite graph. All shortest s-t-paths using an edge e of the $(h-1)$th bipartite graph can be constructed by selecting one vertex of each of the first $h-2$ triples and one edge adjacent to e of the rightmost bipartite graph. Therefore all edges sharing one vertex of the hth triple have the same shortest path frequency. These values are listed for the first six iterations in Table 7.1, where the rightmost column applies to the case $h = 2$ only. To find a most cost effective edge we have to consider the cheapest edge of each group only. The corresponding ratios are given in Table 7.2. Furthermore, the right column shows the ratio of the most cost effective edge in the rightmost bipartite graph as mentioned above in (ii). Thus, in the ith iteration

It.\ c_e	$\{1/9, 1/8, 1/7\}$	$\{1/6, 1/5, 1/4\}$	$\{1/3, 1/2, 1/1\}$
1	3	3	3
2	2	3	3
3	2	2	3
4	2	2	2
5	1	2	2
6	1	1	2

Table 7.1. Shortest path frequencies of the edges of the $(h-1)$th bipartite graph divided by 3^{h-2}.

It.\ c_e	$1/9$	$1/6$	$1/3$	max. $f(e)/c_e$ in the hth $K_{3,3}$
1	9	6	3	9
2	6	6	3	8
3	6	4	3	7
4	6	4	2	6
5	3	4	2	5
6	3	2	2	4

Table 7.2. Ratios $f(e)/c_e$ of selected edges of the $(h-1)$th and hth bipartite graph divided by 3^{h-1}.

the edge with cost $1/(10-i)$ of the rightmost bipartite graph is most cost effective. After the sixth iteration an instance equivalent to the one with $h-1$ bipartite graphs remains and the assertion follows by induction.

The total cost of the edges deleted by the greedy algorithm is

$$\begin{aligned} C_{\text{greedy}} &= h\left(\frac{1}{9}+\frac{1}{8}+\frac{1}{7}+\frac{1}{6}+\frac{1}{5}+\frac{1}{4}\right)+\left(\frac{1}{3}+\frac{1}{2}+1\right) \\ &= \frac{2509}{2520}h+\frac{11}{6}. \end{aligned}$$

The number of vertices of these instances is $n = 3h+6$ leading to

$$C_{\text{greedy}} = \frac{2509}{7560}n - \frac{199}{1260} \geq 0.331n - 0.158.$$

Since the total cost of the unique optimal solution is $C_{\text{opt}} = 1$ the theorem is proved. The decisions made by the greedy algorithm are not

unique but clearly this can be achieved by slightly modifying the edge costs, for example by increasing the cost of the edges of the rth bipartite graph by $(h-r)\varepsilon$ and by increasing the cost of $\{w,t\}$ by $h\varepsilon$ for small $\varepsilon > 0$. ❑

Clearly, the graph used to prove this theorem is not planar since it contains a $K_{3,3}$. However, a similar construction with a planar graph is possible. The graph consists of $h+1$ pairs of vertices instead of triples of vertices, see Figure 7.4, and can be drawn without intersecting edges as shown for $h = 6$ in Figure 7.5. Edge costs as shown in Figure 7.6

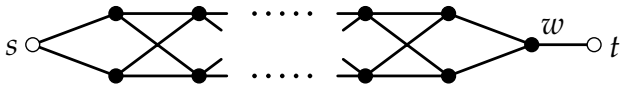

Figure 7.4. Bad planar instances for the greedy algorithm.

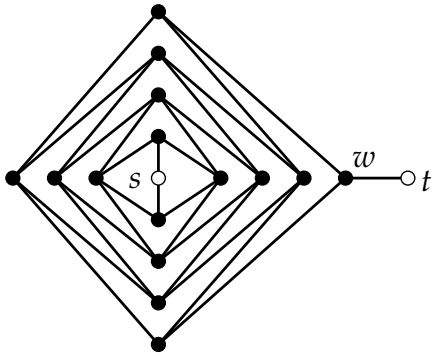

Figure 7.5. Planar drawing of the graph in Figure 7.4 with $h+1 = 7$ pairs.

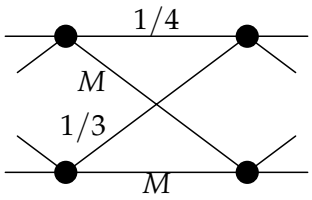

Figure 7.6. Edge costs of the instance in Figure 7.4.

are applied to the bipartite graphs, except for the leftmost part where the costs M are replaced by $1/2$ and 1. The edge $\{w,t\}$ gets cost 1, all other edges get cost M. Then by similar arguments as in the previous

proof one can show that the greedy algorithm possibly deletes all edges having costs smaller than M except for $\{w,t\}$. This solution has total cost

$$C_{\text{greedy}} = h\left(\frac{1}{4}+\frac{1}{3}\right)+\left(\frac{1}{2}+1\right) = \frac{7}{12}h+\frac{3}{2}.$$

Using the equality $n = 2h+5$ we get

$$C_{\text{greedy}} = \frac{7}{12}\left(\frac{n-5}{2}\right)+\frac{3}{2} = \frac{7}{24}n+\frac{1}{24}.$$

Since the optimal objective value is 1, we have the following result.

Theorem 7.4
For instances of BSP with a planar graph on n vertices and one pair the greedy algorithm may determine a solution that is

$$\frac{7}{24}n+\frac{1}{24} \geq 0.291n+0.041$$

times as expensive as an optimal solution.

Even if the graph has a simpler structure no sub-linear approximation ratio can be guaranteed as the following theorem shows.

Theorem 7.5
For instances of BSP with a grid graph on n vertices and one pair the greedy algorithm may determine a solution that is

$$(n+4)/6 \geq 0.166n+0.666$$

times as expensive as an optimal solution.

PROOF. Consider instances consisting of a sequence of r squares, any two consecutive squares sharing one vertex, and one additional edge as shown in Figure 7.7 for $r = 4$. One edge of each square has cost 1,

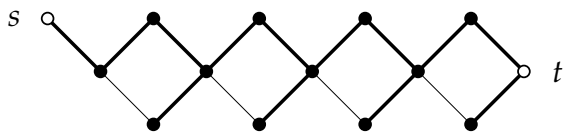

Figure 7.7. Bad instances with a grid graph for the greedy algorithm. Thick edges have cost 2, thin edges have cost 1.

all other edges have costs 2. In the first iteration the edge adjacent to

s has a shortest s-t-path frequency of 2^r, for all other edges this value is 2^{r-1}. Therefore the leftmost thin edge may be removed. The remaining graph is equivalent to an instance with $r-1$ squares and it follows by induction that all thin edges and finally one thick edge may be removed. The objective value of this solution is $r+2 = (n+4)/3$, where $n = 3r+2$ is the number of vertices. Since the optimal objective value is 2, the approximation ratio is $(n+4)/6$. ❑

The examples in the previous proofs are a bit unsatisfying since the solution determined by the greedy algorithm is not *minimal*, that is, it can easily be improved by excluding superfluous edges. This technique is called *pruning*. In fact, for the instance in the proof of Theorem 7.3 this may lead to a solution with total cost $7129/2520 \leq 2.83$ for all n. So it is an obvious question what is the best possible approximation guarantee if we apply this kind of post processing. At least, $2\ln((n-3)/2)$ is a lower bound on this value. This can easily be seen by examples with an odd number n of vertices as shown in Figure 7.8 for $n = 11$. They consist of a complete bipartite graph with $h = (n-3)/2$ vertices

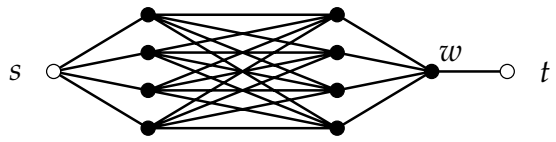

Figure 7.8. Bad instances for the greedy algorithm with pruning.

in each set. All vertices of one set are adjacent to s, the vertices of the other set are adjacent to a vertex w which in turn is adjacent to t. The edge $\{w,t\}$ has cost $1+\varepsilon$ and forms the unique optimal solution. The edges of the complete bipartite graph have costs

$$\frac{1}{1}, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \ldots, \frac{1}{h^2}$$

in arbitrary order. All other edges have costs 2. The shortest path frequency of $\{w,t\}$ is equal to the number of edges in the bipartite graph, namely $h^2 - i$ after the ith iteration. The greedy algorithm will delete all edges of the bipartite graph in order of increasing costs. All decisions are unique and the determined solution is minimal. For $\varepsilon \to 0$ the approximation factor is

$$\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots + \frac{1}{h^2} = H_{h^2} \geq \ln(h^2) = 2\ln\left(\frac{n-3}{2}\right),$$

where H_i is the ith harmonic number. Whether or not the enhanced greedy algorithm gives just a linear approximation for some instances of BSP remains open.

7.3 LP based rounding

In this section we discuss an approximation algorithm using LP rounding techniques. It is an application on the approximation for VERTEX COVER and SET COVER problems proposed by Hochbaum in [33]. We use the MIP defined on page 51:

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e & & \\ \text{s. t.} \quad & x_{\{u,v\}} + y^i_u - y^i_v \geq 0 & & \text{for all } (u,v) \in D_{s_i} \\ & & & \text{and all } i = 1, \ldots, k \\ & y^i_{s_i} = 0 & & \text{for all } i = 1, \ldots, k \\ & y^i_{t_i} = 1 & & \text{for all } i = 1, \ldots, k \\ & x_e \in \{0,1\} & & \text{for all } e \in E \end{aligned} \tag{7.1}$$

LP rounding can be applied to this formulation as follows.

Algorithm 7.3 (LP Rounding)
Determine $d = \max\{\text{dist}(s_i, t_i) \mid i = 1, \ldots, k\}$. Then find an optimal solution (x^*, y^*) of the relaxation of system (7.1) with $x_e \geq 0$ instead of $x_e \in \{0,1\}$ for all $e \in E$ and return the edge set

$$S = \{e \in E \mid x^*_e \geq 1/d\}.$$

Clearly, this algorithm has polynomial running time and as the following theorem shows its output is a feasible solution with approximation guarantee d.

Theorem 7.6
Algorithm 7.3 determines a feasible solution of BSP that is at most d times as expensive as an optimal solution.

PROOF. To see that the solution S determined by Algorithm 7.3 is feasible let P be a shortest s_i-t_i-path for some i. If we add up the inequalities $x_e + y^i_u - y^i_v \geq 0$ for all $e = \{u,v\} \in P$ then the variables of the interior

vertices of P cancel out. Therefore, after setting $y^i_{s_i} = 0$ and $y^i_{t_i} = 1$ we have

$$x_{e_1} + x_{e_2} + \ldots + x_{e_r} \geq 1,$$

where $e_1, \ldots, e_r$ are the edges of P. So there is an index i with $x_{e_i} \geq 1/r$. Since $1/r \geq 1/d$ we have $e_i \in S$ and deleting S from G blocks P.

The total cost of the edges in S is at most d times as large as the optimal solution of the LP-relaxation. Since the optimal objective value of the relaxation is not larger than the optimal objective of system (7.1) the total cost of S is in turn at most d times as large as an optimal solution of BSP. ❑

If not all considered distances are the same then we can improve Algorithm 7.3 by determining for each edge e an individual bound

$$d_e = \max\{\text{dist}(s_i, t_i) \mid e \text{ is an edge of a shortest } s_i\text{-}t_i\text{-path}\}$$

to decide whether to round up or down. Then the solution is

$$S = \{e \in E \mid x^*_e \geq 1/d_e\}.$$

Nevertheless, in general the approximation guarantee given in Theorem 7.6 is tight, as the following theorem shows.

Theorem 7.7
For instances of BSP with grid graphs and unit costs Algorithm 7.3 may determine a solution which is d times as expensive as an optimal solution where $d = \max\{\text{dist}(s_i, t_i) \mid i = 1, \ldots, k\}$ is the maximum considered distance.

PROOF. To prove this assertion we consider instances of BSP with an $(h+1) \times (h+1)$-grid graph with unit costs (see Figure 7.9 for $h = 5$). Any two vertices having distance h form one pair, so the maximal distance is $d = h$. Thus, the horizontal edges being in one row form a shortest path of some pair. The same is true for the vertical edges being in one column. Since these $2(h+1)$ paths all are disjoint, a feasible solution consists of at least $2(h+1)$ edges. Since the grid graph contains $2h(h+1)$ edges this bound is tight for the solution $(\widehat{x}, \widehat{y})$ with $\widehat{x}_e = 1/h$ for all $e \in E$ and $\widehat{y}^i_v = \text{dist}(s_i, v)/h$.

To show that $(\widehat{x}, \widehat{y})$ is also a basic solution we prove that if $(\widehat{x}, \widehat{y}) + (x, y)$ and $(\widehat{x}, \widehat{y}) - (x, y)$ both are feasible points then $(x, y) = 0$ follows. The proof is by contradiction, so assume that both points are feasible and $(x, y) \neq 0$. All inequalities are tight for $(\widehat{x}, \widehat{y})$, therefore the condition $A\binom{x}{y} = 0$ must hold where A is the matrix of the relaxation of

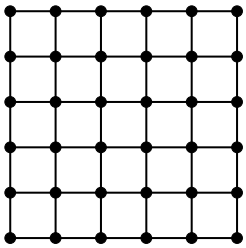

Figure 7.9. A 6×6-grid graph.

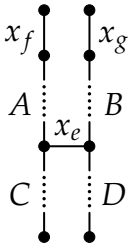

Figure 7.10.

system (7.1). Clearly, $y^i_{s_i} = y^i_{t_i} = 0$ holds. If we add up the inequalities of the edges of a shortest path P as in the proof of Theorem 7.6 we get the condition

$$\sum_{e \in P} x_e = 0.$$

In other words x is an assignment of values to the edges such that for each considered shortest path the sum of the numbers assigned to the edges is 0.

Assume that $x \neq 0$. Then there is an edge $e = \{u, v\}$ with $x_e \neq 0$. After an appropriate rotation of the grid e is a horizontal edge not belonging to the top row. Then we consider the vertical paths connecting u and v to the second row from the top and to the bottom row of the grid. Let A, B, C, and D be the sums of the assigned numbers along these paths or 0 if the paths contain no edges, see Figure 7.10. The missing edges in the two columns are denoted by f and g, respectively. Since the vertical edges of one column form a shortest path we have

$$(x_f + A + C) + (x_g + B + D) = 0.$$

On the other hand we have two shortest paths using e and $h - 1$ vertical edges starting at the bottom row of the grid. This leads to

$$(A + x_e + D) + (B + x_e + C) = 0.$$

The difference of the two equalities is $x_f + x_g - 2x_e = 0$. Since $x_e \neq 0$ we have

$$x_f + x_g \neq 0. \tag{7.2}$$

Now we consider a shortest path consisting of the vertical edges of one column. We can replace the topmost edge by a horizontal edge going

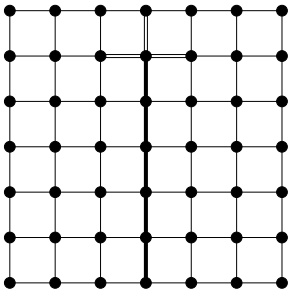

Figure 7.11. The double edges all have the same assigned numbers.

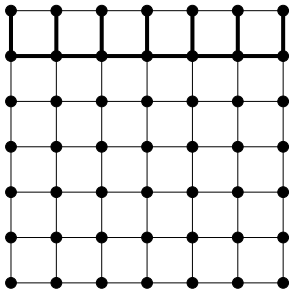

Figure 7.12. The value 0 is assigned to all thick edges.

rightwards or leftwards (if they exist), see Figure 7.11. As this is also a shortest path the sum of the assigned numbers is 0 for all three (two) paths. Since they differ in one edge only, the topmost vertical edge and the two (one) adjacent horizontal edges in the second top row all have the same assigned value. Because this is true for all columns of the grid, the horizontal edges in the second top row and the vertical edges connecting the top and the second top row all have the same value (see Figure 7.12). Because the edges of the second row from the top form a shortest path the numbers assigned to all these edges including x_f and x_g must be 0, a contradiction to (7.2).

Hence, $x = 0$ holds. From $A\binom{x}{y} = 0$ and $y^i_{s_i} = y^i_{t_i} = 0$ we get $y = 0$. Therefore, $(\widehat{x}, \widehat{y})$ is an optimal basic solution and may be determined by Algorithm 7.3. As a result of rounding all edges of the grid are deleted. The solution determined by Algorithm 7.3 therefore may have $2h(h+1)$ edges.

The following integer solution for odd h has an objective value of $2(h+1)$ and is therefore optimal. It consists of the middle column of horizontal edges and of the middle row of vertical edges, see Figure 7.13. Deleting these edges leaves four vertex-disjoint grid graphs with $(h+1)/2 \times (h+1)/2$ vertices. Thus, no shortest path of length h remains. The approximation ratio then is $h = d$.

If h is even, then there is an integer solution consisting of $1 + h/2$ horizontal edges in the $(h/2)$th column and $1 + h/2$ horizontal edges in the $(1 + h/2)$th column starting at the bottom row and the top row, respectively. Furthermore, it consists of $1 + h/2$ vertical edges in the $(h/2)$th row and $1 + h/2$ vertical edges in the $(1 + h/2)$th row starting

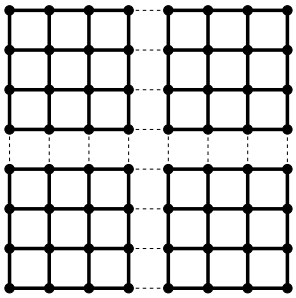

Figure 7.13. Optimal solution for $h = 7$.

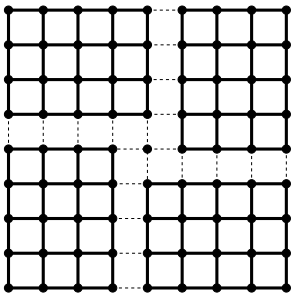

Figure 7.14. Optimal solution for $h = 8$.

at the leftmost column and the rightmost column, respectively. Then four vertex-disjoint grid graphs with $(h/2) \times (h/2+1)$ vertices remain which do not contain shortest paths of length h, see Figure 7.14. The objective value is $2h+4$.

To see that this solution is actually optimal consider the four shortest paths starting at the central vertex going $h/2$ edges rightwards or $h/2$ edges leftwards and then $h/2$ edges upwards or $h/2$ edges downwards, see Figure 7.15. We already mentioned that at least one edge

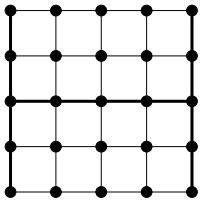

Figure 7.15. These shortest paths between the central vertex and the vertices on the corners cannot be blocked by one edge per row and one edge per column.

has to be deleted in each row of horizontal edges and in each column of vertical edges. Furthermore we have to delete two horizontal edges in the middle row or we have to delete two vertical edges of the rightmost column or two vertical edges of the leftmost column to block these four paths. In either case one additional edge has to be deleted. If we apply the same argumentation to the graph rotated by 90 degrees then we get another edge that has to be deleted additionally. This gives a lower bound of $2(h+1)+2 = 2h+4$ proving the asserted optimality.

Hence, we get

$$\frac{2h(h+1)}{2h+4} \to h$$

as the approximation ratio for even h, completing the proof. ❑

As in the example for the greedy algorithm in the previous section the solution determined by the approximation algorithm is not minimal. So the solution may be improved by excluding redundant elements in a random order as before. However, unlike the former case the approximate solution of this example gives no information, so instead of pruning the solution one could start from scratch and guess a solution of the BSP instance. Also for *iterative rounding*, that is, iteratively fixing the largest component of x to 1 and considering the reduced problem, no better performance guarantee can be proved, since values of x can be arbitrarily small.

7.4 Dual approach

Another way to get a factor d approximation for BSP, where again

$$d = \max\{\text{dist}(s_i, t_i) \mid i = 1, \ldots, k\},$$

is the SET COVER approximation presented by Bar-Yehuda and Even in [5]. Its application to BSP is as follows.

Algorithm 7.4
Find an unblocked shortest path P and determine its cheapest edge e. Decrease costs of all edges of P by c_e and then delete e from G. Iterate until all shortest s_i-t_i-paths are blocked.

For each path P that was explicitly blocked by Algorithm 7.4 by deleting its cheapest edge e, let y_P be the current weight of e at the moment of deletion. For all other shortest paths P let $y_P = 0$. Then it is easy to see that y is a feasible solution of the dual problem of the LP-relaxation of (6.1) on page 49. Furthermore, it can be proved that the total cost of the determined solution is at most d times the objective value of y, see [5]. Therefore Algorithm 7.4 is a factor d approximation. Examples whose graphs are grid graphs and caterpillars as shown in Figure 7.16 prove that the approximation guarantee d is tight. If the pair (s_1, t_1) is considered first then one of the horizontal edges having unit costs is deleted and the costs of the remaining horizontal edges are set to

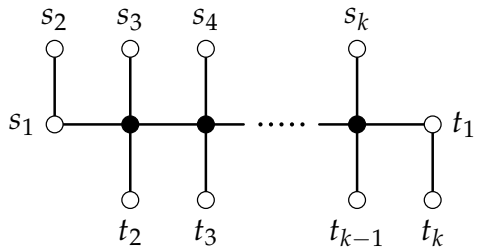

Figure 7.16. Bad instances for Algorithm 7.4, vertical edges have costs $\varepsilon > 0$ and horizontal edges have unit costs.

0. This deletion blocks the shortest path of one additional pair. Iteratively, for each of the other pairs the horizontal edge of its shortest paths is deleted. So Algorithm 7.4 deletes all unit cost edges, while all optimal solutions consist of one horizontal edge and $k-2$ vertical edges if $\varepsilon < 1$. The approximation factor therefore is

$$\frac{k-1}{1+(k-2)\varepsilon} \to k-1 = d$$

for small ε, proving the following theorem.

Theorem 7.8
For instances of BSP Algorithm 7.4 may determine a solution which is $d = \max\{\text{dist}(s_i, t_i) \mid i = 1, \ldots, k\}$ *times as expensive as an optimal solution even if the graph is a caterpillar or a grid graph.*

Observe that all decisions made by Algorithm 7.4 are unique if (s_1, t_1) is considered first. Furthermore, the determined solution is minimal, that is, no edge can be excluded from the solution without losing feasibility.

It was shown that the performance of Algorithm 7.4 is not better than the one of the LP rounding algorithm described in the previous section. Admittedly, the former algorithm is simple to implement and has a favorable complexity: Initially, the distances of the considered pairs can be determined using k breadth-first-searches. Then we need just one breadth-first-search for each deleted edge plus one breadth-first-searches for each $i = 1, \ldots, k$ showing that all shortest s_i-t_i-paths are blocked. So at most $2k + m = \mathcal{O}(n^2)$ breadth-first-searches are necessary, the running time therefore is $\mathcal{O}(n^2 m)$.

7.5 Summary

In this chapter we studied approximation algorithms with performance guarantees

- $\min\{k, n-1\}$,
- $d = \max\{\text{dist}(s_i, t_i) \mid i = 1, \ldots, k\}$,
- $\ln k + n \ln 3/3 + 1 - 2 \ln 3/3$.

So if k or d is relatively small then one would use the decomposition algorithm or LP based rounding, respectively. Otherwise, if k and d both are larger than about $n/3$ then the greedy algorithm seems to be more appropriate. Of course this rule may be suboptimal in some cases. For even h consider the bad planar instance for the greedy algorithm with $h^2/2 - 1$ bipartite blocks, see Figure 7.4. Starting at t we insert a path with $h^2/2 - 1$ edges all having cost M, and then we add vertices $t_1, \ldots, t_h$ and $s_1, \ldots, s_h$ and connect them to the other end vertex of these paths and to s, respectively, by edges with costs M, see Figure 7.17. Consider the BSP instance with this graph and pairs (s_i, t_j),

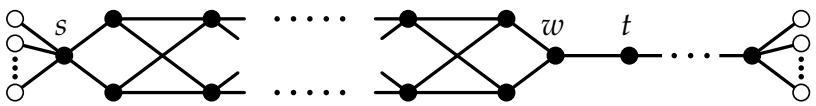

Figure 7.17. Theoretically bad instances for LP based rounding and decomposition.

$i, j = 1, \ldots, h$. We have

$$\begin{aligned} n &= h + 1 + 2\frac{h^2}{2} + 1 + \frac{h^2}{2} + h = \frac{3}{2}h^2 + 2h + 2, \\ k &= h^2, \\ d &= 2 + \frac{h^2}{2} - 1 + 2 + \frac{h^2}{2} - 1 + 1 = h^2 + 3. \end{aligned}$$

The approximation guarantee of the greedy algorithm then is

$$\ln(k) + 0.366n - 0.268 = \ln(h^2) + 0.549h^2 + 0.732h + 0.464$$

which is smaller than k and smaller than d for $h \geq 4$ suggesting to prefer this algorithm. Indeed, since all shortest path frequencies are

just multiplied by h^2 we have the worst case behavior as shown before, whereas decomposition and LP based rounding determine the optimal solution as one can easily prove.

As mentioned earlier all approximation algorithms can be enhanced by simple pruning. For this purpose edges of the solution are considered in a random order. For each of them it is checked whether it can be excluded from the solution without affecting feasibility. If this is the case then the solution is modified, in either case the next edge is tested. In Chapter 8 there is a comparison of the approximation ratios of the four presented algorithms with and without pruning when applied to random instances.

Another important aspect of an approximation algorithm is its running time. If an LP solver is available then both, decomposition and LP rounding are easy to implement and the running time is quite short. Also the dual algorithm is fast, at most $2k + m$ breadth-first-searches are needed. For the greedy algorithm the running time may be much longer since we determine the current shortest path frequencies in each iteration. The straightforward approach is to determine this value for each pair separately as described in the proof of Lemma 7.1 using two breadth-first-searches among others. This results in $2k$ breadth-first-searches for each deleted edge, this takes a long time for the instances discussed in Chapter 8. Of course the effort can be reduced since the shortest path frequency $f_{s,t}$ is changed only if the deleted edge is an edge of a shortest s-t-path. Therefore this value has to be recomputed only for some pairs. Nevertheless, the greedy algorithm had the longest running time in the computational experiments presented in the next chapter.

CHAPTER 8

Computational Results

BESIDES the theoretical analysis of the complexity of a problem in combinatorial optimization one should always consider the actual hardness for *real world* sized instances. Clearly, in some cases a polynomial time algorithm may have a long running time while an exponential time approach may be quite reasonable to solve average instances. An example for such a case is LINEAR PROGRAMMING with the *theoretically fast* ellipsoid method and the *theoretically slow* simplex methods. Similar, approximation algorithms may determine solutions of a good quality in many cases, while only for a small set of instances the performance is bad.

The algorithms for BSP presented in this work were tested with five sets of 100 random instances each. In this chapter we describe generation, preprocessing, and computational results of these BSP instances. To present the resulting values and to give an impression of their distribution the tables show for each quantity and each set the minimum (min), the average (avg), the maximum (max), and the standard deviation (dev).

All computations were performed on a 2.8 GHz Pentium IV PC with 1 GB of memory running Linux 2.4.20 and ILOG CPLEX 9.0. The programs were written in C and compiled with gcc 3.2.2.

8.1 The test instances

For the computational experiments we consider *Erdős-Rényi-graphs*, $G(n, p)$, that is, graphs with n vertices whose edges are selected independently with the same probability p. The edge costs are integers uniformly distributed in some given range. The pairs are disjoint, their

vertices are selected randomly. We created 100 instances for each of five sets of parameters, namely the number n of vertices, the number k of pairs, the edge probability p, and the cost range. Initially, the following simple preprocessing is applied to the instances.

Edges that are not used by one of the considered shortest paths are not part of an optimal solution, so we can exclude them from the graph. Afterwards, isolated vertices also can be taken out. Finally, we search for independent subproblems. More precisely, for each pair in the BSP instance there should be a shortest path using a critical edge (see Definition 6.1 on page 52). Otherwise this subproblem can be solved separately in polynomial time. Therefore such pairs were removed. To get an overview, Table 8.1 shows for each set the initial number n_i of vertices, the initial number k_i of pairs, the edge probability p, and the range of edge costs (c range). Moreover, it shows the number n of vertices, the number m of edges, and the number k of pairs after preprocessing. The two rightmost columns give the average distance of the considered pairs and the number of critical edges in the reduced graph. As one might expect for *Erdős-Rényi-graphs* all these values are in a small range for each set.

8.2 Optimal solutions

To determine optimal solutions we used the MIP in (6.2) on page 51, the computations were performed with CPLEX using default settings. Table 8.2 shows for each set the optimal objective value, the total time for CPLEX computation in seconds, the ratio between optimal objective value of the relaxation and optimal objective value of the MIP, the time to solve the relaxation in seconds, the number of nonzeros after CPLEX's preprocessing, the number of branch-and-cut nodes, and the number of simplex iterations.

The maxima of the computation times are rather large but as shown by the standard deviation there are only a few outliers as usual for random instances of *NP* hard problems. The reason for long computation times are evidently large branch-and-cut trees, the relaxation is solvable in less than a second. Also important for short computation times is the quality of the bound determined by the relaxation. For BSP the bounds seem to be better if distances are small.

Apparently, MIPs of unit cost instances are harder to solve using CPLEX. Especially between set A and set B there is a big difference

set	parameters for instance generation				characteristics after preprocessing				
	n	k	p	c range	n	m	k	average distance	critical edges
A	5000	800	0.001	1...1	3644	6210	750	5	2114
					3753.3	6614.9	769.0	5.5	2376.2
					3860	6988	783	6	2694
					40.8	135.9	5.1	0.03	106.5
B	5000	800	0.001	50...55	3628	6045	753	5	2038
					3747.0	6596.6	767.8	5.5	2368.7
					3857	7018	782	6	2618
					38.8	129.2	4.8	0.03	101.4
C	8000	2000	0.00016	1...1	1477	1519	239	21	875
					1791.0	1854.2	328.4	25.4	1153.6
					2060	2185	402	31	1400
					105.4	114.3	26.7	1.55	83.8
D	8000	2000	0.00016	50...55	1356	1390	224	21	824
					1754.1	1816.0	320.8	25.4	1132.1
					2267	2379	449	33	1465
					126.6	135.9	33.3	1.48	96.0
E	100000	1800	0.000015	1...1	10726	11318	559	23	3099
					11327.8	12035.7	607.2	24.3	3554.2
					12042	12870	657	25	3892
					258.2	283.9	15.4	0.28	131.4

Table 8.1. Characteristics of the initial and the preprocessed test instances. In the five rightmost columns each cell gives minimum value, average value, maximum value, and standard deviation.

although costs in the second set are in a small range and therefore should change optimal solutions only slightly. For set C and set D the difference is smaller but still evident. This behavior is typical of unit cost MIPs in general. It is assumed that this is a consequence of the symmetry in the branch-and-bound tree. This problem can be got under control by slightly modifying the costs, the difference may be well smaller then for the set B.

The number of critical edges should be an indicator for the computation time since the corresponding variables are the one to branch on, but counterintuitively the average solution time for set A well exceeds the one for set F. This holds also for sets D and B.

8.3 Approximation algorithms

Solutions for the instances were determined with the approximation algorithms presented in Chapter 7, namely decomposition, greedy, LP based rounding, and the dual approach. In a second step a pruning was performed, that is, superfluous edges were removed iteratively. Table 8.3 shows the approximation ratios without and with pruning for each algorithm. Although time was not exactly measured it may be remarked that the greedy algorithm was clearly slower than the three others.

Two small modifications were applied to the algorithms which give no theoretical improvement but may increase practical performance. For the decomposition algorithm edges of the solution of the current subproblem were removed from the graph before solving the next subproblem. Especially if optimal solutions of the subproblems are not unique or if there are several nearly optimal solutions then this may be helpful. For the LP based rounding individual bounds were used for each edge to decide whether or not to select it. This bound is the reciprocal of the longest of the considered paths using this edge, see discussion after Theorem 7.6 on page 64. This is only advantageous if the distances of pairs are quite different. For the random instances under consideration this is probably not the case.

At first glance the deviations of the approximation ratios are small in most cases, especially after pruning. So algorithms seem to behave predictable on most random instances. For our instances the pruning step is worthwhile at most for LP rounding and the dual approach. For the decomposition the benefit is rather small and for the greedy algo-

rithm the additional effort seems no to be maintainable.

One would expect the decomposition algorithm to be better at sets C and D than at the others because of their relatively small numbers of pairs. However, this is not the case even after pruning. Another characteristic that should influence the approximation ratio of this algorithm is the number of critical edges, if it tends to 0 then decomposition should be almost optimal. Nevertheless, this value is relatively small for sets C and D again. In fact, the path length seems to be the critical value. If we consider unit cost instances and random cost instances separately then we see the ratio increasing with the average distance.

The greedy algorithm performs good for all instances and the approximation ratios are close to each other, so no evident rule can be seen from these results.

For LP based rounding the expectation one might have after the theoretical analysis proves true in these experiments. The algorithm performs significantly better for sets A and B with small distances than for the remaining sets with large distances. The only exceptional fact is that the ratio for set E is better than the one for sets C and D although the average distance is nearly the same and the other characteristics should make the instances of the former set harder.

For the dual approach also the approximation ratio increases with the average distance, although things are not as evident as for LP rounding. Unlike the latter algorithm, the dual approach without and with pruning gives nearly the same ratios for sets C and D as one might expect.

Comparing the practical results one would suggest to use the LP rounding with pruning if distances are small. It is simple to implement if an LP solver is available, has a short running time, and it performs better than the other algorithms for sets A and B. If the paths are long as in sets C, D, and E then the greedy algorithm has the best performance but its running time is quite long. In such cases one has to evaluate this quality/speed tradeoff carefully.

set	opt	CPX-time in sec.	relaxation ratio	relax-time in sec.	nonzeros	b&b nodes	simplex iterations
A	676	3.1	0.9882	0.1	6611	2	2421
	713.8	2627.7	0.9923	0.3	7955.3	68275.1	3824876.7
	754	117663.6	0.9957	0.8	9701	2641017	138312051
	13.0	4352.1	0.0013	0.10	565.3	105712.0	6054863.9
B	46586	0.3	0.9936	0.1	7159	0	1662
	49972.4	10.5	0.9966	0.1	9016.4	384.3	12680.7
	52467	102.9	0.9985	0.2	10354	6727	192615
	955.1	8.5	0.0007	0.03	470.5	465.9	13181.5
C	27	0.4	0.8859	0.0	3560	0	643
	41.9	91.7	0.9242	0.0	5598.1	12986.1	462136.5
	59	5943.9	0.9694	0.1	7149	687912	26478904
	4.4	146.7	0.0115	0.02	546.2	19690.9	727274.6
D	1816	0.9	0.9141	0.0	4303	4	384
	2650.3	59.1	0.9476	0.1	7232.0	6685.5	201323.8
	3714	1707.6	0.9776	0.1	9931	126293	5448175
	307.3	81.4	0.0102	0.02	854.4	8971.5	281458.4
E	230	2.9	0.9782	0.1	5151	42	3137
	247.1	794.0	0.9851	0.2	6038.7	40082.8	2078731.2
	272	11919.4	0.9925	0.5	6764	770229	44302010
	6.4	1003.9	0.0023	0.08	288.1	50681.6	2684015.2

Table 8.2. Objective values, problem sizes, and computation times of the MIP formulations. Each cell gives minimum value, average value, maximum value, and standard deviation. All times are in seconds.

set	decomposition		greedy		LP rounding		dual	
A	1.36	1.27	1.07	1.07	1.11	1.01	1.61	1.16
	1.40	1.31	1.10	1.10	1.34	1.04	1.67	1.23
	1.45	1.34	1.12	1.12	1.51	1.07	1.73	1.26
	0.013	0.010	0.008	0.008	0.055	0.008	0.019	0.012
B	1.22	1.17	1.04	1.04	1.07	1.01	1.59	1.15
	1.26	1.19	1.05	1.05	1.24	1.03	1.66	1.18
	1.29	1.23	1.07	1.06	1.42	1.06	1.71	1.22
	0.010	0.010	0.004	0.004	0.057	0.009	0.017	0.010
C	1.75	1.44	1.08	1.08	1.50	1.06	1.70	1.18
	2.11	1.67	1.15	1.14	2.61	1.19	2.04	1.37
	3.35	1.91	1.29	1.27	3.17	1.31	2.26	1.53
	0.137	0.079	0.029	0.028	0.232	0.046	0.092	0.056
D	1.52	1.27	1.04	1.04	1.09	1.01	1.59	1.14
	1.79	1.42	1.12	1.11	2.48	1.20	1.76	1.29
	2.14	1.68	1.21	1.20	3.07	1.34	2.04	1.45
	0.096	0.077	0.026	0.023	0.213	0.045	0.069	0.049
E	1.83	1.63	1.10	1.09	1.39	1.06	1.94	1.32
	1.95	1.73	1.13	1.13	1.84	1.11	2.03	1.38
	2.07	1.88	1.17	1.17	2.09	1.16	2.12	1.43
	0.037	0.036	0.011	0.011	0.110	0.017	0.032	0.018

Table 8.3. Approximation ratios for the presented algorithms without and with pruning. Each cell gives minimum value, average value, maximum value, and standard deviation.

CHAPTER 9

Final Remarks

In this chapter we present variations of the problem discussed in this work. For some of them results can be adopted, while for others obvious questions remain open.

9.1 The directed case

For BSP we considered undirected graphs only, but clearly the problem can be defined for digraphs as well. We denote this variation by BSPd. Note that pairs of BSPd instances are *ordered* pairs, that is, $(s,t) \neq (t,s)$. In fact, the directed case is a generalization since each edge $\{u,v\}$ of an undirected graph can be replaced by a copy of a special digraph given by Baier in [4], see Figure 9.1. This transformation

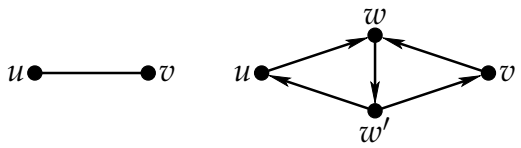

Figure 9.1. Edges of a BSP instance can be replaced by a copy of this digraph. We set $c_{(w,w')} = c_{\{u,v\}}$, all other edges get costs $c_{\{u,v\}} + 1$.

triples lengths of paths, and consequently there is a directed shortest path in the resulting digraph for any shortest path in the original graph, and vice versa. Deleting an edge $\{u,v\}$ corresponds exactly to deleting the edge (w,w') in its replacement. Hence, the resulting BSPd instance is equivalent to the original BSP instance.

Many results in this work are true for BSPd as well. Theorem 4.1 on page 30 states that BSP can be solved in polynomial time if there is one vertex s occurring in all k pairs. For BSPd we get the following result.

Theorem 9.1
BSPd with pairs (s,t_i), $i = 1,\ldots,k$ or with pairs (t_i,s), $i = 1,\ldots,k$ can be solved in polynomial time.

PROOF. To solve corresponding BSP instances one has to construct the shortest paths digraph with root s, see Definition 4.1 on page 29. It contains those edges with an appropriate direction that are edges of considered shortest paths. This can be done similarly for BSPd instances: We ignore all edges (u,v) for which $\operatorname{dist}(s,v) \neq \operatorname{dist}(s,u)+1$ holds and determine a minimal s-T-cut in the remaining graph, where $T = \{t_1,\ldots,t_k\}$. Thus, instances where only shortest paths starting at s are considered can be solved in polynomial time.

If all directed edges in the graph as well as the elements of each pair are flipped, then the set of shortest paths is not changed and feasibility of edge sets is preserved. Therefore this operation does not change the BSPd instance in principal and the directed variation can also be solved in polynomial time if only shortest paths ending at s are considered. ❑

Before we discuss the case where shortest paths starting at s and shortest paths ending at s both are considered, we present the analog of Corollary 4.1 and Corollary 4.2 (see pages 33 and 34).

Theorem 9.2
BSPd is NP-hard even for $k = 2$ pairs and grid graphs with maximum degree $\Delta = 3$, and for $k = 2$ pairs and planar graphs with unit costs.

PROOF. To prove the assertion one cannot use the transformation described at the beginning of this section since this would violate the properties of bounded maximum degree and of unit costs. However, one can use the same reduction as in the proof of Theorem 4.2 and assign directions to the edges as shown in Figure 9.2. Edges outside the cells get the obvious directions going from s_i to the cells and from the cells to t_i. For the modification in the proof of Corollary 4.2 edges of the 8-cycles are oriented counter clockwise. The variation used to prove Corollary 4.1 can also be applied in the directed case. ❑

The just described directed version of the reduction can be modified slightly to get additional results.

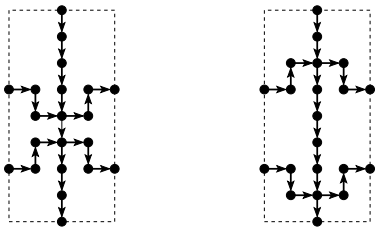

Figure 9.2. Tiles of type 1 (left) and type 2 (right) with directed edges.

Theorem 9.3
BSPd is NP-hard even for grid graphs with pairs (s,r), (r,t), *with pairs* (r,r'), (r',r), *and for all triangle instances, that is, instances where the edges corresponding to the three pairs form a triangle.*

PROOF. We use the reduction provided in the previous proof and we insert an additional vertex r, a directed t_2-r-path, and a directed r-s_1-path, see Figure 9.3. The edges of these paths get sufficiently large

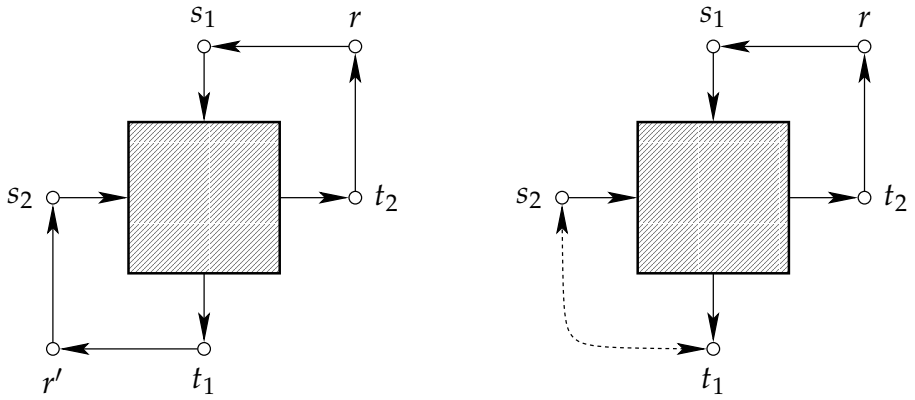

Figure 9.3. Modifications of the reduction proving *NP*-hardness for instances of BSPd with certain pair configurations.

costs ($c+1$, see proof of Theorem 4.2) such that they cannot be deleted. These additional paths do not change the set of considered shortest paths. Now we consider the BSPd instance with this graph and vertex pairs (s_2,r) and (r,t_1). Any solution with total cost at most c is also feasible for the instance with pairs (s_1,t_1) and (s_2,t_2), and vice versa. Therefore this problem is *NP*-hard as well.

Now we insert another vertex r' and a directed t_1-r'-path and a directed r'-s_2-path with large edge costs. Consequently, the original BSPd

instance is equivalent to the BSPd instance with pairs (r, r') and (r', r). Therefore the latter one is *NP*-hard.

Consider the graph with the additional vertex r but without r'. In each triangle configuration there is one vertex being the first vertex of one pair and the second vertex of another pair. Therefore, only the configurations

$$(s, r), \quad (r, t), \quad (s, t)$$

and

$$(s, r), \quad (r, t), \quad (t, s)$$

have to be considered. In both cases we assign $s = s_2$ and $t = t_1$. In the first case we insert a directed s-t-path going downwards and rightwards only, in the second case we insert a directed t-s-path going leftwards and upwards only. The costs of these edges are 0. In both cases the additional paths are the unique shortest path between s and t and they are not influencing the reduction. Hence, triangle instances of BSPd are *NP*-hard. ❑

In the reductions described so far, we can replace each directed edge e with cost c_e by a complete bipartite graph K_{2,c_e} with edges having an appropriate orientation as in the proof of Corollary 4.1 on page 33. This leads to the following result.

Corollary 9.1
BSPd is NP-hard even for planar bipartite graphs with two pairs. This is also true for pairs (s, r), (r, t), and for pairs (r, r'), (r', r), and for all triangle instances, that is, instances where the edges corresponding to the three pairs form a triangle.

9.2 Increasing distances by exactly 1

The problem of blocking shortest paths discussed throughout this work asks for a minimum cost edge set whose deletion increases $\mathrm{dist}(s_i, t_i)$ for each i by at least 1. We did not care about the exact value by which $\mathrm{dist}(s_i, t_i)$ increases. So what about seeking a minimum cost edge set whose deletion increases $\mathrm{dist}(s_i, t_i)$ by *exactly* 1 for each i? The problem can be defined as follows.

EXACT DISTANCE INCREASING (EDI).
Given an undirected graph $G = (V, E)$ with edge costs c_e and

k vertex pairs (s_i, t_i), find a minimum cost edge set S such that

$$\text{dist}_{G \setminus S}(s_i, t_i) = \text{dist}_G(s, t) + 1$$

for all $i = 1, \ldots, k$.

Given an instance of BSP, we could insert additional vertices and edges forming an s_i-t_i-path of length $\text{dist}(s_i, t_i) + 1$ with sufficiently large edge costs to achieve an instance of EDI. Then any feasible solution of the original BSP instance is also feasible for the constructed EDI instance, and vice versa. Therefore EDI is as hard to solve as BSP. Since this reduction preserves objective values of solutions there is also no *PTAS* for EDI, see Theorem 4.5 on page 39. In this section we discuss the problem of finding a feasible solution of EDI instances. A trivial necessary condition for solvability of EDI instances is that there are s_i-t_i-paths of length $\text{dist}(s_i, t_i) + 1$ for each $i = 1, 2, \ldots, k$. This rules out bipartite graphs since edges of a shortest s-t-path and edges of an s-t-path with length $\text{dist}(s, t) + 1$ form an odd cycle. As the following theorem shows it is even *NP*-hard to decide whether a feasible solution exists.

Theorem 9.4
Given an instance of EDI it is NP-complete to decide whether a feasible solution exists, even if the graph is planar.

PROOF. First we observe that this problem is in *NP* since verifying a solution can be done by k breadth-first-searches.

For the completeness we consider the *NP*-complete problem EXACT COVER BY 3-SETS. We are given sets

$$A_1, A_2, \ldots, A_m \subseteq A = \{1, 2, \ldots, 3n\}$$

with $|A_i| = 3$ and ask for an index set I with $|I| = n$ and

$$\bigcup_{i \in I} A_i = A.$$

Note that as a consequence of the cardinalities a solution of this problem actually is a partition of A, that is, each element occurs in exactly one of the selected sets. An EDI instance to solve this problem can be constructed as follows. For each set A_i we take a path with vertices $v_{i,1}, v_{i,2}, \ldots, v_{i,6n+1}$. Each edge of this path is augmented to a triangle using an additional vertex, see Figure 9.4 for $n = 1$. Then we add pairs $(v_{i,j}, v_{i,j+2})$ for $j = 1, \ldots, 6n - 1$ to the instance, that is, each two vertices of the primal path with distance 2 form one pair. Consequently,

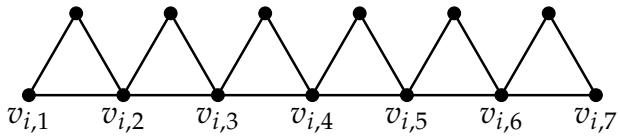

Figure 9.4. The row corresponding to the set A_i in the proof of Theorem 9.4 for $n = 1$.

there are two possibilities to increase the distances of these pairs by exactly 1: One has to delete the edges $\{v_{i,j}, v_{i,j+1}\}$ either for all odd $j = 1, 3, \ldots, 6n-1$ or for all even $j = 2, 4, \ldots, 6n$. This corresponds to selecting A_i or not selecting A_i, respectively. To ensure that the selected sets make an exact cover we insert additional pairs $(v_{0,2j}, v_{m,2j})$ for $j = 1, 2, \ldots, 3n$, where $v_{0,2j}$ are supplementary vertices. These pairs correspond to the elements of A. For all i and j the vertices $v_{i,2j-1}$ and $v_{i-1,2j}$ are connected by an edge if $j \in A_i$. Otherwise the vertices $v_{i,2j}$ and $v_{i-1,2j}$ are connected by a path of length 2, see Figure 9.5. Notice

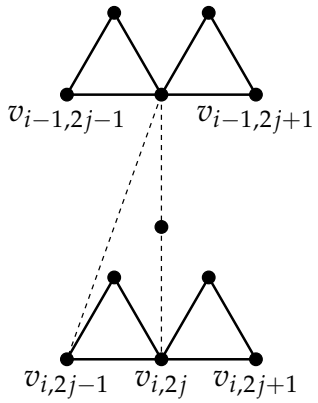

Figure 9.5. Example for the connection of neighboring rows in the proof of Theorem 9.4. Exactly one of the dashed paths is included in the graph.

that $\operatorname{dist}(v_{0,2j}, v_{i,2j}) = 2i$, whereas $\operatorname{dist}(v_{0,2j}, v_{i,2k}) \geq 2i + 2$ for $k \geq j + 1$. Now consider the $3n$ columns each containing two triangles per row. All $v_{0,2j}$-$v_{m,2j}$-paths of length at most $\operatorname{dist}(v_{0,2j}, v_{m,2j}) + 1$ are completely in the jth column. For each row in which the odd edges are deleted and for which the corresponding set contains j the value $\operatorname{dist}(v_{0,2j}, v_{m,2j})$ is increased by exactly 1. Deleting edges between vertices of different rows increases this distance by at least 2. So these edges are not part of a feasible solution. Consequently, a solution is feasible if and only if

there is for each j exactly one index i such that in the ith row the odd edges are deleted and such that $j \in A_i$. Hence, increasing the distances by exactly 1 is equivalent to a solution of the given covering problem.

Clearly, the presented graph is planar and its size and the time to construct it is bounded polynomially by the size of the EXACT COVER instance. ❑

So we showed that finding a feasible solution of EDI is *NP*-complete if the number of pairs is not fixed. Trivially, for one pair this problem is easy since we only have to find an edge $\{u, v\}$ such that

$$\text{dist}(s, u) + 1 + \text{dist}(v, t) = \text{dist}(s, t) + 1$$

holds. Deleting all edges except for $\{u, v\}$ and the edges of the two corresponding shortest paths is a feasible solution. If no such edge exists then there is no s-t-path of length $\text{dist}(s, t) + 1$. Whether or not this problem can be solved in polynomial time if the number of pairs is a constant $k \geq 2$ remains open.

9.3 Increasing distances by at least $d \geq 2$

In the previous section we thought of BSP as the problem of increasing distances. This view leads to the following generalization.

> **DISTANCE INCREASING PROBLEM (DIP).**
> Given an undirected graph $G = (V, E)$ with edge costs c_e for all $e \in E$ and vertex pairs (s_i, t_i), $i = 1, \ldots, k$, find a minimum cost edge set S such that
>
> $$\text{dist}_{G \setminus S}(s_i, t_i) \geq \text{dist}_G(s, t) + d$$
>
> for all $i = 1, \ldots, k$.

As mentioned in Chapter 2 we set $\text{dist}(s, t) = \infty$ if there is no s-t-path. For $d = 1$ DIP is the same as BSP. For fixed $d \geq 2$ this problem is *NP*-hard for $k = 2$ pairs, as the following theorem shows.

Theorem 9.5
For fixed $d \geq 2$ DIP is NP-hard, even

(i) for $k = 2$ pairs and grid graphs with maximum degree 3,

(ii) for $k = 2$ pairs and planar bipartite graphs with unit cost edges.

Furthermore, DIP is APX-hard in stars with unit costs.

PROOF. To prove this we can reduce BSP to DIP. For this purpose we replace any edge of a given BSP instance by a path of length d whose edges all have the same costs as the original edge. As a result, the length of each s_i-t_i-path is multiplied by d, and blocking shortest paths is equivalent to increasing distances by at least d. Since this reduction does not change the properties planar, bipartite, unit costs, maximum degree, and grid graph, DIP is also *NP*-complete in the stated cases, as was shown for BSP in Corollary 4.1 (page 33) and in Corollary 4.2 (page 34), respectively. Furthermore, the reduction used to prove Theorem 4.5 (page 39) can also be applied to the case $d \geq 2$. ❑

After this simple reduction only the complexity of instances with $k = 1$ pair remains open. Unlike BSP there is no strong duality for DIP in this case, that is, the maximum number of edge disjoint paths that have to be blocked may be smaller than the minimal number of edges in a feasible solution. To see this consider instances as shown for $d = 3$ in Figure 9.6. These instances consist of sequences of $2d - 1$ graphs $K_4 - e$. Any two consecutive $(K_4 - e)$s share a vertex. For an

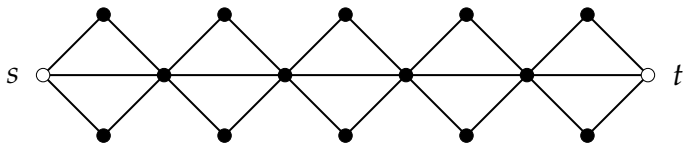

Figure 9.6. An instance of DIP with $d = 3$ and unit cost edges. The optimal objective value is 2 but no two of the paths to block are edge-disjoint.

s-t-path P let h be the number of $(K_4 - e)$s whose horizontal edges are not part of P. Then the length of P is $\text{dist}(s,t) + h$ and therefore this path has to be blocked only if $h \leq d - 1$. Consequently, the s-t-paths to be disconnected have at least d horizontal edges and no two of them are edge disjoint. However, there is evidently no feasible solution with just one edge. Dropping the unit cost property we could assign costs M to the non-horizontal edges. Then the objective value of the dual problem is unchanged while all optimal solutions of the DIP instance are formed by d horizontal edges, giving a ratio of d between the optimal objective values of the primal and the dual problem.

A straightforward approach to solve instances of DIP with $k = 1$ pair and some d is to increase $\text{dist}(s,t)$ iteratively by (at least) 1 until a total advance of d is achieved. Of course, this may be suboptimal as

can be proved using instances as shown in Figure 9.7 for $d = 3$. These

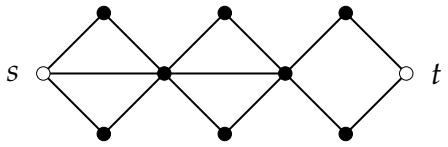

Figure 9.7. An instance of DIP with $d = 3$. Horizontal edges have costs 2, all other edges have costs $1 + \varepsilon$.

instances are similar to those described before, but they have only d blocks and the rightmost block has no horizontal edge. The horizontal edges have costs 2, all other edges have costs $1 + \varepsilon$. It is easy to see that using the proposed approach, in each iteration one horizontal edge and finally a cut is deleted yielding an objective value of

$$2(d-1) + 2(1+\varepsilon) = 2(d+\varepsilon).$$

Deleting the two edges adjacent to t is an optimal solution with objective value $2(1+\varepsilon)$, so the approximation ratio converges to d as $\varepsilon \to 0$. In fact, this is the worst case behavior, since an optimal solution of the DIP instance contains a feasible solution for the at most d subproblems (compare the proof of Theorem 7.1).

If the integer d is part of the input data then this problem is *APX*-complete even for $k = 1$ pair and unit costs as Baier showed, see Corollary 2.15 on page 41 in [4].

Despite these partial results, the question whether DIP with $k = 1$ pair can be solved in polynomial time for some fixed d remains open.

Bibliography

[1] Noga Alon, András Gyárfás, and Miklós Ruszinkó, *Decreasing the diameter of bounded degree graphs*, J. Graph Theory **35** (2000), no. 3, 161–172.

[2] G. Anandalingam and S. Raghavan (eds.), *Telecommunications network design and management*, Operations Research/Computer Science Interfaces Series, vol. 23, Kluwer Academic Publishers, Boston, MA, 2003.

[3] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi, *Complexity and approximation*, Springer-Verlag, Berlin, 1999.

[4] Georg Baier, *Flows with path restrictions*, Ph.D. thesis, Technische Universität Berlin, 2003.

[5] Reuven Bar-Yehuda and Shimon Even, *A linear-time approximation algorithm for the weighted vertex cover problem*, J. Algorithms **2** (1981), no. 2, 198–203.

[6] Richard Bellman, *On a routing problem*, Quart. Appl. Math. **16** (1958), 87–90.

[7] Marshall Bern and Paul Plassmann, *The Steiner problem with edge lengths* 1 *and* 2, Inform. Process. Lett. **32** (1989), no. 4, 171–176.

[8] Daniel Bienstock and Nicole Diaz, *Blocking small cuts in a network, and related problems*, SIAM J. Comput. **22** (1993), no. 3, 482–499.

[9] Abdelmadjid Bouabdallah, Charles Delorme, and Selma Djelloul, *Edge deletion preserving the diameter of the hypercube*, Discrete Appl. Math. **63** (1995), no. 1, 91–95.

[10] Andreas Brandstädt, Van Bang Le, and Jeremy P. Spinrad, *Graph classes: a survey*, SIAM Monographs on Discrete Mathematics and Applications, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1999.

[11] Gary Chartrand and Frank Harary, *Planar permutation graphs*, Ann. Inst. H. Poincaré Sect. B (N.S.) **3** (1967), 433–438.

[12] Nicos Christofides, *Worst-case analysis of a new heuristic for the travelling salesman problem*, Algorithms and Complexity: New Directions and Recent Results (J.F. Traub, ed.), Academic Press, 1976, p. 441.

[13] John Clark and Derek Allan Holton, *A first look at graph theory*, World Scientific Publishing Co. Inc., Teaneck, NJ, 1991.

[14] Charles J. Colbourn, *The combinatorics of network reliability*, International Series of Monographs on Computer Science, The Clarendon Press, Oxford University Press, New York, 1987.

[15] Stephen A. Cook, *The complexity of theorem-proving procedures.*, Proc. 3rd ann. ACM Sympos. Theory Computing, Shaker Heights, Ohio 1971, ACM, New York, 1971, pp. 151–158.

[16] Elias Dahlhaus, David S. Johnson, Christos H. Papadimitriou, Paul D. Seymour, and Mihalis Yannakakis, *The complexity of multiterminal cuts*, SIAM J. Comput. **23** (1994), no. 4, 864–894.

[17] E. W. Dijkstra, *A note on two problems in connexion with graphs*, Numer. Math. **1** (1959), 269–271.

[18] Ehab S. El-Mallah and Charles J. Colbourn, *The complexity of some edge deletion problems*, IEEE Trans. Circuits and Systems **35** (1988), no. 3, 354–362.

[19] István Fáry, *On straight line representation of planar graphs*, Acta Univ. Szeged. Sect. Sci. Math. **11** (1948), 229–233.

[20] L. R. Ford, Jr. and D. R. Fulkerson, *Flows in networks*, Princeton University Press, Princeton, N.J., 1962.

[21] Lester R. Ford Jr. and Delbert R. Fulkerson, *Maximal flow through a network*, Canad. J. Math. **8** (1956), 399–404.

[22] Toshihiro Fujito, *A unified approximation algorithm for node-deletion problems*, Discrete Appl. Math. **86** (1998), no. 2-3, 213–231.

[23] Toshihiro Fujito, *Approximating node-deletion problems for matroidal properties*, J. Algorithms **31** (1999), no. 1, 211–227.

[24] Michael R. Garey and David S. Johnson, *Computers and intractability*, W. H. Freeman and Co., San Francisco, CA, 1979.

[25] Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis, *Primal-dual approximation algorithms for integral flow and multicut in trees*, Algorithmica **18** (1997), no. 1, 3–20.

[26] Niall Graham and Frank Harary, *Changing and unchanging the diameter of a hypercube*, Discrete Appl. Math. **37/38** (1992), 265–274.

[27] M. Grötschel, L. Lovász, and A. Schrijver, *The ellipsoid method and its consequences in combinatorial optimization*, Combinatorica **1** (1981), no. 2, 169–197.

[28] Martin Grötschel, Laszlo Lovász, and Alexander Schrijver, *Polynomial algorithms for perfect graphs*, Topics on perfect graphs, North-Holland Math. Stud., vol. 88, North-Holland, Amsterdam, 1984, pp. 325–356.

[29] U. I. Gupta, D. T. Lee, and J. Y.-T. Leung, *Efficient algorithms for interval graphs and circular-arc graphs*, Networks **12** (1982), no. 4, 459–467.

[30] Frank Harary, *Graph theory*, Addison-Wesley Publishing Co., Reading, Mass.-Menlo Park, Calif.-London, 1969.

[31] Frank Harary and Robert W. Robinson, *The diameter of a graph and its complement*, Amer. Math. Monthly **92** (1985), no. 3, 211–212.

[32] Johan Håstad, *Some optimal inapproximability results*, STOC '97 (El Paso, TX), ACM, New York, 1999, pp. 1–10.

[33] Dorit S. Hochbaum, *Approximation algorithms for the set covering and vertex cover problems*, SIAM J. Comput. **11** (1982), no. 3, 555–556.

[34] Frank K. Hwang, Dana S. Richards, and Pawel Winter, *The Steiner tree problem*, Annals of Discrete Mathematics, vol. 53, North-Holland Publishing Co., Amsterdam, 1992.

[35] Bernhard Korte and Jens Vygen, *Combinatorial optimization*, second ed., Algorithms and Combinatorics, vol. 21, Springer-Verlag, Berlin, 2002.

[36] Peter Kubat and J. MacGregor Smith (eds.), *Special issue: Topological network design in telecommunication systems*, Ann. Oper. Res. **106** (2002), 345.

[37] Kazimierz Kuratowski, *Sur le probléme des courbes gauches en topologie*, Fund. Math. **15** (1930), 217–283.

[38] John M. Lewis and Mihalis Yannakakis, *The node-deletion problem for hereditary properties is NP-complete*, J. Comput. System Sci. **20** (1980), no. 2, 219–230.

[39] Michael Okun and Amnon Barak, *A new approach for approximating node deletion problems*, Inform. Process. Lett. **88** (2003), no. 5, 231–236.

[40] Christos H. Papadimitriou and Kenneth Steiglitz, *Combinatorial optimization: algorithms and complexity*, Dover Publications Inc., Mineola, NY, 1998.

[41] Panos M. Pardalos and Dingzhu Du (eds.), *Network design: connectivity and facilities location*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 40, American Mathematical Society, Providence, RI, 1998.

[42] S. B. Rao, N. M. Singhi, and K. S. Vijayan, *The minimal forbidden subgraphs for generalized line-graphs*, Combinatorics and graph theory (Calcutta, 1980), Lecture Notes in Math., vol. 885, Springer, Berlin, 1981, pp. 459–472.

[43] A. A. Schoone, H. L. Bodlaender, and J. van Leeuwen, *Diameter increase caused by edge deletion*, J. Graph Theory **11** (1987), no. 3, 409–427.

[44] Alexander Schrijver, *Combinatorial optimization. Polyhedra and efficiency*, Algorithms and Combinatorics, vol. 24, Springer-Verlag, Berlin, 2003.

[45] Tomomi Segawa, *Radius increase caused by edge deletion*, SUT J. Math. **30** (1994), no. 2, 159–162.

[46] Douglas R. Shier, *Network reliability and algebraic structures*, Oxford Science Publications, The Clarendon Press Oxford University Press, New York, 1991.

[47] Alfonso Shimbel, *Structural parameters of communication networks*, Bull. Math. Biophys. **15** (1953), 501–507.

[48] J. MacGregor Smith and P. Winter (eds.), *Topological network design*, Baltzer Science Publishers BV, Bussum, 1991.

[49] Vijay V. Vazirani, *Approximation algorithms*, Springer-Verlag, Berlin, 2001.

[50] Ľubomír Šoltés, *Forbidden induced subgraphs for line graphs*, Discrete Math. **132** (1994), no. 1-3, 391–394.

[51] Klaus Wagner, *Bemerkung zum Vierfarbenproblem*, Jber. Deutsch. Math.-Verein. **46** (1936), 26–32.

[52] Douglas B. West, *Introduction to graph theory (2nd edition)*, Prentice Hall Inc., Upper Saddle River, NJ, 2001.

[53] Yuansheng Yang, Jianhua Lin, and Chunli Wang, *Three forbidden subgraphs for line graphs*, Discrete Math. **252** (2002), no. 1-3, 287–292.

[54] Mihalis Yannakakis, *Edge-deletion problems*, SIAM J. Comput. **10** (1981), no. 2, 297–309.

[55] Mihalis Yannakakis, Paris C. Kanellakis, Stavros S. Cosmadakis, and Christos H. Papadimitriou, *Cutting and partitioning a graph after a fixed pattern*, Automata, languages and programming, 10th Colloq., Barcelona/Spain 1983, Lect. Notes Comput. Sci. 154, Springer-Verlag, Berlin, 1983, pp. 712–722.

Name Index

Subject Index

List of Symbols and Abbreviations

(u, v)	directed edge with first vertex u and second vertex v, see page 5
$\{u, v\}$	undirected edge with vertices u and v, see page 5
$[a, b]$	closed interval $\{x \in \mathbb{R} \mid a \leq x \leq b\}$
$\overline{G}$	Complement of a graph G, see page 6
$G \cong G'$	Graphs G and G' are isomorphic, see page 6
APX	class of *approximable problems*, see page 14
BSP	BLOCKING SHORTEST PATHS, see page 2
BSPd	BLOCKING SHORTEST PATHS IN DIGRAPHS, see page 81
BSPv	BLOCKING SHORTEST PATHS BY VERTEX DELETION, see page 22
$c(e)$	cost of edge e
C_n	cycle with n vertices, see page 8
$\deg_G(v)$	degree of vertex v in G, see page 6
$\operatorname{diam}(G)$	diameter of a graph G, see page 7
DIP	DISTANCE INCREASING PROBLEM, see page 87
$\operatorname{dist}_G(u, v)$	distance of vertices u and v in G, see page 7
EDI	EXACT DISTANCE INCREASING, see page 84
$f_{s,t}(e)$, $f(e)$	shortest path frequency of edge e, see page 55
$G(n, p)$	Erdős-Rényi-graph, see page 73
H_i	ith harmonic number, see page 63
K_n	complete graph with n vertices, see page 7
$K_{s,t}$	complete bipartite graph with s and t vertices, see page 8
$\ln x$	natural logarithm of x
LP	linear program, see page 15
MIP	mixed integer program, see page 15
$\mathbb{N}$	the set of natural numbers $\{1, 2, \ldots\}$

NP	class of *non-deterministic polynomial solvable problems*, see page 12
P	class of *polynomial solvable problems*, see page 12
P_n	path with n vertices, see page 8
PSP	PACKING SHORTEST PATHS, see page 43
PTAS	class of problems for which a *polynomial time approximation scheme* exists, see page 14
$\mathbb{Q}$	the set of rational numbers
Q_n	n-dimensional hypercube, see page 10
$\mathbb{R}$	the set of real numbers
$\mathbb{R}_+$	the set of non-negative real numbers $\{x \in \mathbb{R} \mid x \geq 0\}$
$\mathrm{rad}(G)$	radius of a graph G, see page 7
TSP	TRAVELING SALESMAN PROBLEM, see page 14
$\mathbb{Z}$	the set of integers $\{0, 1, -1, 2, -2, \ldots\}$
$\mathbb{Z}_+$	the set of non-negative integers $\{0, 1, 2, \ldots\}$
$\chi(G)$	chromatic number of G, see page 22
$\delta_G(S)$	the cut defined by the vertex set S in the graph G, see page 7
$\delta(G)$	minimum degree of G, see page 7
$\Delta(G)$	maximum degree of G, see page 7

Zusammenfassung

In der vorliegenden Arbeit wird ein spezielles SET COVER Problem studiert, das bislang in der Literatur nicht diskutiert wurde. Es ist eng mit minimalen Schnitten in Graphen verknüpft. Im Unterschied zu letztgenannten Problemen wird bei dem hier betrachteten nicht gefordert, alle Wege zwischen den Knoten eines oder mehrerer gegebener Knotenpaare zu unterbrechen, sondern es reicht aus, die kürzesten Wege zu durchtrennen. Mit anderen Worten besteht die Aufgabe darin, in einem ungerichteten Graphen den Abstand zwischen den Knoten gegebener Paare durch Entfernen einer kostenminimalen Kantenmenge um mindestens 1 zu erhöhen. Das Problem wird als BLOCKING SHORTEST PATHS oder kurz BSP bezeichnet.

Nach der Einleitung werden im zweiten Kapitel grundlegende Begriffe und Zusammenhänge der Gebiete Graphentheorie, Komplexitätstheorie und Diskrete Optimierung, die in dieser Arbeit benötigt werden, zusammengestellt. In Kapitel 3 werden dann verwandte Probleme, die in der Literatur behandelt wurden, sowie bekannte Resultate dazu kurz vorgestellt.

Die Komplexität von BSP wird in Kapitel 4 betrachtet. Dabei werden bezüglich der Anzahl der Knotenpaare, des Graphen und der Kosten spezielle Fälle diskutiert und jeweils *NP*-Vollständigkeitsbeweise oder polynomielle Algorithmen angegeben. Lediglich für den Fall von BSP mit drei Paaren, die ein Dreieck bilden, bleibt der *P*/*NP*-Status offen. Im fünften Kapitel werden nicht-algorithmische Fragen behandelt. Dabei geht es um Abschätzungen des optimalen Zielfunktionswertes von Instanzen mit Einheitskosten, um die Dualitätslücke, sowie um die Anzahl der kürzesten Wege zwischen zwei Knoten eines Graphen.

In Kapitel 6 werden zwei Ansätze zur Lösung von BSP mittels ganzzahliger linearer Optimierung vorgestellt. Der erste ist die naheliegende SET COVER basierte Formulierung, der zweite basiert auf der LP-Formulierung für das MINIMUM CUT Problem. Zur Approximation von BSP werden in Kapitel 7 vier Standardtechniken angewandt. Die bestmögliche Approximationsgarantie wird in drei Fällen exakt be-

stimmt und im vierten Fall relativ eng eingegrenzt.

Praktische Ergebnisse, die für zufällige BSP Instanzen die Rechenzeiten und die Approximationsgüte zeigen, werden in Kapitel 8 vorgestellt. Im letzten Kapitel werden schließlich naheliegende Erweiterungen und Variationen der betrachteten Fragestellung studiert. Es handelt sich um das BSP entsprechende Problem in gerichteten Graphen, um das Problem die Abstände um *genau* 1 zu erhöhen, sowie um das Problem die Abstände um mindestens $d \geq 2$ zu erhöhen. Es werden jeweils *NP*-Vollständigkeitsbeweise angegeben.